AFOSR-TR-97-0653

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank)*

2. REPORT DATE
1 August 1997

3. REPORT TYPE AND DATES COVERED
Final Technical report, 8/1/94 to 7/31/97

4. TITLE AND SUBTITLE
Mathematical Theory of Neural Networks

5. FUNDING NUMBERS
AFOSR-94-0293
F49620-94-1-0293

6. AUTHORS
Eduardo Sontag
Hector Sussmann

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Rutgers, The State University of New Jersey
New Brunswick, NJ 08903

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
AFOSR/NM, Building 410
Bolling AFB, DC 20332-6448

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release
distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report focuses on fundamental theoretical issues relevant to the capabilities, performance, and limitations of artificial neural networks. For static (feedforward) networks, subjects of investigation included the study of error surfaces for least squares fitting, VC and other learning dimensions, representability questions, and function approximation. For dynamic (recurrent) nets, covered are questions dealing with parameter identification and modeling, realizability and other systems-theoretic issues, theoretical computational capabilities, and learning-theoretic issues.

14. SUBJECT TERMS
neural networks
learning
systems theory

15. NUMBER OF PAGES
39

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
UNCLASSIFIED

20. LIMITATION OF ABSTRACT

MATHEMATICAL THEORY OF NEURAL NETWORKS

Eduardo D. Sontag and Héctor J. Sussmann
SYCON – Rutgers Center for Systems and Control
Department of Mathematics, Rutgers University
New Brunswick, NJ 08903

ABSTRACT

This report focuses on fundamental theoretical issues relevant to the capabilities, performance, and limitations of artificial neural networks.

For static (feedforward) networks, subjects of investigation included the study of error surfaces for least squares fitting, VC and other learning dimensions, representability questions, and function approximation. For dynamic (recurrent) nets, covered are questions dealing with parameter identification and modeling, realizability and other systems-theoretic issues, theoretical computational capabilities, and learning-theoretic issues.

1 Introduction

This project focused on theoretical issues regarding capabilities, performance, and limitations of "neural computation." (Artificial) neural networks are structures inspired by the way biological nervous systems process information. They consist of a large number of highly interconnected relatively simple processing elements or "neurons" whose collective behavior is interpreted as a computation, typically involving tasks of classification, function evaluation, systems simulation, or control. Most often, neural networks are designed by numerical parameter-fitting routines, such as gradient descent, which are in part appealing because of their analogies to "learning" in biological systems by means of adjustments to (excitatory or inhibitory) synaptic connections. Much research carried out by AFOSR contractors and Air Force labs concerns the use of neural network techniques when dealing, among others, with problems of fault detection and classification (in particular, for reconfigurable aircraft), design of controls valid over large flight envelopes, and precision laying of composites on aircraft structures. Unquestionably, there is a demand for basic theoretical foundations for the analysis and comparison of different models and algorithms. This work concerns the development of such foundations.

The area of neural computing has seen several periods of high interest and expectations, since at least the late 1940s, separated by periods of reaction to exaggerated claims by some of its proponents. During the times of less activity, there was nonetheless a continued research effort by major researchers such as Kohonen and Grossberg all along, but the latest resurgence in interest was due in large part to the two independent developments, namely the introduction and study of:

1. classes of functions defined by *feedforward nets* or "*multilayer perceptrons*" with sigmoidal activations, which are dense (in various spaces of real-valued vector functions), could in principle be fit to data through nonlinear optimization, and may be evaluated by finely-grained parallel computation, and

2. classes of dynamical systems called *feedback* or *recurrent* nets, which also possess approximation properties (now with regards to classes of dynamical systems), parametric forms suitable for optimization, and parallel computation characteristics.

The first line of work was popularized by the Rumelhart and McClelland "PDP" books, while the second was heavily influenced by a technique for associative memory storage and retrieval due to Hopfield. The catalyst for both was the contemporaneous availability of low-cost computing power (for simulations and implementations) made possible by microprocessor technology, in amounts which were unimaginable when similar ideas had been considered in the past.

1.1 Main Motivation for This Work

Many practical successes of the associated technologies have been claimed in both the engineering and popular press. A fair amount of research exists by now on theoretical issues for neural networks, though papers are scattered in various journals. The "NeuroCOLT" project in Europe is one excellent source of material which is available online, as well as references. (See also the first PI's Web pages, and references later and in cited papers, for further pointers to the literature.) There are several textbooks dealing with theoretical issues concerning neural networks, such as the one recently published by Vidyasagar, but a great deal of basic questions are still poorly understood.

1.2 Another Motivation

There is another motivation as well which underlies the work reported here. This arises from an issue that has come up in many domains –including computer, communications, and control engineering– concerning the interface between on the one hand the continuous. physical. world and on the other hand discrete devices such as digital computers, capable of symbolic processing. Lately the term "hybrid" systems has become popular in this context.

For instance, classical control techniques, especially for linear systems, have proved spectacularly successful in automatically regulating relatively simple systems; however, for large-scale problems, controllers resulting from the application of the well-developed theory are used as building blocks of more complex systems. The integration of these systems is often accomplished by means of ad-hoc techniques that combine pattern recognition devices, various types of switching controllers, and humans –or, more recently, expert systems,– in supervisory capabilities. This has caused renewed interest in the formulation of mathematical models in which the interface between the continuous and the symbolic is naturally accomplished and system-theoretic questions can be formulated and resolved for the resulting models. Successful approaches will eventually allow the interplay of modern control theory with automata theory and other techniques from computer science. This interest has motivated much research into areas such as discrete-event systems, supervisory control, and more generally "intelligent control systems". As an illustration of the interest level this subject attracts, suffice it to say that a recent workshop at Rutgers, co-sponsored and cofunded by the PIs' Center for Systems and Control (SYCON), was attended by well over a hundred researchers in hybrid systems (only a subset of those who applied, being constrained by space and funding limitations), and resulted in a lively exchange of ideas, reflected in the extensive proceedings volume [6]. (The PIs are actively involved in organization activities for further hybrid systems meetings, including membership in the Grenoble 1997 program committee.)

The present work has as one of its underlying objectives the analysis of neural networks as a paradigm in which to understand hybrid systems issues. Other paradigms could be used as well; it so turns out that neural nets appear to be a particularly appealing and very mathematically natural class of nonlinear systems, as will be explained in the report.

Similarly, in numerical analysis and optimization, much activity has taken place in the realm of continuous algorithms. Included here are such areas as differential-equation implementations of "interior methods" for linear and nonlinear programming, the use of flows on manifolds to solve eigenvalue and optimization problems (in particular the work of Brockett and his school), and the Blum-Shub-Smale approach to "real valued" algorithms. All these deal in one way or another with the power of "analog computing" to solve problems that can also be attacked with discrete/symbolic techniques. Again, we view the neural net paradigm as one in which to explore the interface between "digital" and "analog" modes of computation. In fact, work by one of the PIs and one of his students has succeeded in formulating a new computer-science approach to these issues and has been reported upon to the general scientific community (*Science*, April 28, 1995).

1.3 Neural Networks

The term "neural network" tends to be applied loosely, making the area too broad and ill-defined. For a mathematical study. especially when providing comparative results, precise definitions are required. For purposes of this report, therefore, we adopt the most popular paradigm: (artificial) neural nets are systems composed of saturation-type nonlinearities, linear elements, and optionally dynamic components (integrators in continuous time, delays in discrete time). When restricting attention to these, while still a huge subject, the scope leaves out a variety of topics sometimes included, such as "radial basis" networks or multiplicative effects (high-order networks); many of our results and reported ideas do apply to such more general classes as well (for instance, those on VC dimension, the estimates on classification of sets in general position, results on impossibility of feedback control when using certain architectures, and so forth), but by narrowing-down the area we can focus our discussion better. There are two basic types of networks: feedforward and feedback, introduced next.

Feedforward Networks

This model consists of feedback-free interconnections of basic units, or "neurons". Each unit operates on some linear combination of the outputs of other units and signals arising from outside the network (the environment). The output produced by each unit has an intensity that is a nonlinear function of this aggregate input, and it is transmitted to other neurons. A subset of the outputs, or a set of linear combinations of these outputs, represents the output of the network as a response to the inputs from the environment. In this manner, a neural network determines an input/output mapping. When viewing the linear combination coefficients as parameters to be "tuned" or adjusted, networks play the role of parametric families of functions.

An often-mentioned engineering justification for the use of neural networks lies in their distributed character. which in principle allows a degree of fault tolerance. Moreover, direct hardware implementations of neural networks exploit their inherent parallelism, and often run orders of magnitude faster than software simulations. Most major chip manufacturers have offered neural net chips, and various complete neurocomputers are commercially available. As one example of direct AF interest, an 8,000-neuron chip developed by AAC (Accurate Automation Corporation) has been reportedly used to control a subscale model of the the USAF unmanned 5.5 Mach "waverider" (cf. *Aviation Week & Space Tech*, April 3, 1995, pp. 78-79).

Also motivating the use of nets is the belief that in some sense they are an especially appropriate family of parameterized models, as opposed to, say, finite Fourier series or splines. Numerical and statistical advantages are said to include excellent capabilities for learning, adaptation, and generalization. We do not argue the case for or against this belief here. Notwithstanding some popular claims to the contrary (for instance, supported by the rate of approximation results reviewed in the report), we feel that the situation is still very unclear regarding the relative merits of using neural nets. Nevertheless, since techniques based on neural networks play a role as part of the general set of tools in estimation, learning, and control, a deeper understanding of the topic is a prerequisite to theoretical comparisons.

Feedback Nets

Recurrent neural networks are those which include dynamic elements (delay lines or integrators, in discrete and continuous time scales respectively), in contrast to feedforward, nets, which only contain static units. Their behavior is described by means of systems of difference (in discrete time) or differential (in continuous time) equations.

Part of this report focuses on such dynamic networks, whose interest arises from the many applications in which input data is a function of time. For example, the inputs fed into a speech recognition system might be sequences consisting of windowed Fourier coefficients and signal levels at each instant; as another example, in control problems the inputs to a regulator might be time-dependent measurements of the plant being controlled as well as the successive coordinates of a path to be tracked. It is desirable to make use of the information implicit in the correlations and dependencies that exist among the inputs at different times, and this can be achieved by considering models which consist of dynamical systems, of which recurrent nets provide a particularly rich subclass.

Recurrent networks are among the models considered by Grossberg and his school during the last twenty or more years, and include the networks proposed by Hopfield for associative memory and optimization. They have been employed in the design of control laws for robotic manipulators (Jordan), as well as in speech recognition (Fallside, Kuhn), speaker identification (Anderson), formal language inference (Giles), and sequence extrapolation for time series prediction (Farmer). In both the areas of signal processing and control, recurrent nets have been proposed as generic identification models or as prototype dynamic controllers. In addition, as discussed later, theoretical results about neural networks established their universality as models for systems approximation as well as analog computing devices. Special purpose chips are being built to implement recurrent nets directly in hardware, just as done for feedforward nets; for instance, Hitachi's Wafer Scale Integration chips have been designed to implement Hopfield nets with over 500 neurons and 30,000 synaptic connections. Electrical circuit implementations of recurrent nets, employing resistively connected networks of nonlinear amplifiers, with the resistor characteristics used to reflect the desired weights, have been suggested as analog computers, in particular for solving constrained optimization problems and for implementing content-addressable memories.

1.4 Learning Theory and Other Methodologies

We wish to study the question *what are special properties of neural networks?* (as models for functions or for dynamical systems). It is extremely difficult to answer directly questions such as "how do nets compare with Fourier expansions?" — obviously, if the task involves periodic signals, it would be innapropriate to use neural networks, but if one has data that is partitioned naturally by affine subspaces, networks may be more indicated. It is more reasonable to search for a theoretical understanding of properties of nets according to various accepted quantifiable criteria, such as rates of approximation, learning-theoretic (VC, Pollard) dimensions, metric entropy of associated function spaces, interpolation and extrapolation ("generalization") measures, pattern classification power, and so forth. This is why we focused much of our work on questions arising from a learning-theoretic framework that allows the formulation of such questions. This framework leads to posing mathematical problems whose solution requires tools from areas as diverse as approximation theory, stochastic processes, or analytic function theory.

Other questions about networks lead to issues of a system-theoretic character, such as deciding if there are redundant internal states in a feedback net (controllability, observability) or if parameters can be identified from input/output data. For these problems, tools from control theory are natural.

1.5 Scope and Organization of the report

The main subtopics of our research can be organized roughly into these broad categories:

- Static ("Feedforward") Nets for Classification and Function Approximation
    - Critical Points of Error Function
    - VC Dimension Bounds

  - Classification of Inputs in General Position
  - Uses for Implementation of State-Feedback Control
    - Single vs Multiple Layers
    - Control of Linear Systems with Saturation: Continuous and Discrete Time
  - Function Approximation

- Dynamic ("Feedback" or "Recurrent") Nets for Systems Modeling
  - Sample Complexity for Identification
  - Parameter Identification and System-Theoretic Aspects
  - Nets as Models of Analog Computing
  - Hybrid Systems: Piecewise Linear

We did not organize this report precisely according to this outline of scope, however. We prefer to follow a more natural discussion, introducing subjects according to their motivation (structure, learning, feedback control, etc). Thus, the first part of this report, Section 2, provides a brief introduction to neural networks, learning theory, construction of feedback, dynamical systems questions, computability, and other ingredients of the work. We attempt to give as few technical details as possible while still conveying the main issues, results, and problems.

Given the range of subjects treated, and the different methodologies employed in each, lack of space prevents giving details on all subtopics. Thus we have chosen only a few selected subtopics, especially those represented in very recent work and hence possibly less easily available to readers, to be explained in somewhat more detail, in the last part, Section 3. Overall, however, most of the discussion in this report is informal, with references to the literature for precise technical points. The Web page

http://www.math.rutgers.edu/~sontag/

provides access to a substantial number of papers by the PIs, and these may be consulted for many more details and a mathematically rigorous presentation.

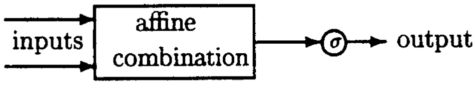
2 Overview

In this part of the report, we discuss the model of neural nets, and we explain in general terms the types of problems considered. The next part has some further technical details on selected subtopics. As mentioned earlier, the Web page found at the URL: **http://www.math.rutgers.edu/~sontag/** provides access to a substantial number of papers by the PIs, to be consulted for details and a mathematically rigorous presentation.

We first discuss the model in informal terms.

Feedforward (Static) Nets

A feedforward neural network is an interconnection of basic processors of a certain particular form.



As discussed in the introductory section, we restrict attention here to the standard paradigm in which each processor is a "first order neuron," though many of our results apply to more general models. A first-order neuron is a device with multiple inputs and a single output channel; it integrates its inputs according to "synaptic" weights (positive values of weights correspond to excitatory connections and negative values to inhibitory ones, in the biological analogy), and, with an intensity that is a function of the aggregate value obtained from this combination of signals, it "fires" its response signal. The inputs to each neuron are, themselves, outputs from other neurons or signals originating from outside the system (inputs to the network). The output signal that a neuron produces is broadcast to the rest of the system. Some set of linear combinations of signals, or as a special case just the outputs of a designated set of "output neurons," is interpreted as the output signal produced by the whole network.

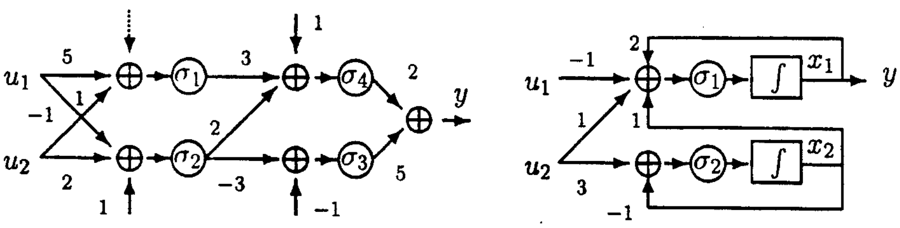
Mathematically, the output produced by each individual neuron in the network has the form $\sigma(a_0 + a_1 u_1 + \ldots + a_m u_m)$, where $(u_1, \ldots, u_m)$ are the inputs to that neuron. The transformation $\sigma : \mathbb{R} \to \mathbb{R}$ is called the "activation function" Sometimes one assumes that all the activations are the same, equal to a fixed σ; when this happens, we'll say here that the net is *homogeneous*. The coefficients a_i of the affine combination of inputs that is computed by each neuron are often called "weights" or "programmable parameters" (a_0 is sometimes called the "bias"), and are in general different for each processor in the given network.

The interconnection structure can be formally specified in terms of a graph whose nodes are the neurons, or equivalently in matrix theoretic terms. Once that this interconnection structure, the activation functions σ for each neuron, and the values of the weights have been fixed, the behavior of the network, as a device transforming external inputs to outputs, is well-defined.

For example, the left part of Figure 1 provides a "block diagram" for a static net which computes the function $y = 2\sigma_4\Big(3\sigma_1(5u_1-u_2)+2\sigma_2(u_1+2u_2+1)+1\Big)+5\sigma_3\Big(-3\sigma_2(u_1+2u_2+1)-1\Big)$. (Ignore both the right diagram and the dotted line for now.) There are four neurons, with activations $\sigma_i, i = 1, 2, 3, 4$, two external inputs u_1 and u_2, and the scalar output y of the network is a linear combination of the outputs of two of the neurons; the numbers $2, 3, \ldots, -1$ are the weights of the network.

The Activation Function

There are several choices for σ that are typical in theory and applications.



$y = 2\sigma_4(3\sigma_1(5u_1 - u_2) + 2\sigma_2(u_1 + 2u_2 + 1) + 1) + 5\sigma_3(-3\sigma_2(u_1 + 2u_2 + 1) - 1)$ $\dot{x}_1 = \sigma_1(2x_1 + x_2 - u_1 + u_2),\ \dot{x}_2 = \sigma_2(-x_2 + 3u_2),\ y = x_1$

Figure 1: *Examples of one Feedforward and one Recurrent Net*

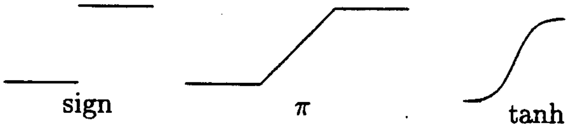


Figure 2: Different Functions σ

The first is the "sign" function, sign $(x) = x/|x|$ (zero for x=0), or its relative, the hardlimiter, threshold, or *Heaviside* function $\mathcal{H}(x)$, which equals 1 if $x > 0$ and 0 for $x \leq 0$ (in either case, one could define the value at zero differently; results do not change much). In order to apply various numerical techniques, one often needs a differentiable σ that somehow approximates sign (x) or $\mathcal{H}(x)$. For this, it is customary to consider the hyperbolic tangent $\tanh(x)$, which is close to the sign function when the "gain" γ is large in $\tanh(\gamma x)$. Equivalently, up to translations and change of coordinates, one may use the ***standard sigmoid***

$$\sigma_s(x) \;=\; \frac{1}{1+e^{-x}}\,. \tag{1}$$

Also common in practice is a piecewise linear function,

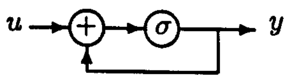
$$\pi(x) = \begin{cases} -1 & \text{if } x \leq -1 \\ 1 & \text{if } x \geq 1 \\ x & \text{otherwise.} \end{cases} \tag{2}$$

This is sometimes called a "semilinear" or "saturated linearity" function.

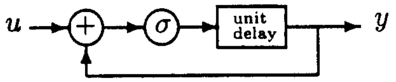
Feedback (Dynamic) Networks

When studying feedforward nets, there is no need to introduce an explicit concept of time. Outputs may be understood either as being produced instantaneously, or after a fixed processing delay, but mathematically it makes no difference: in either case, the behavior of the network can be interpreted as a mapping from input values to output values. We always think of feedforward nets as computing functions between finite-dimensional input and output spaces. As an illustration, the network that was shown in the first example in Figure 1 induces a mapping $\mathbb{R}^2 \to \mathbb{R}$.

When interconnection graphs have cycles, in contrast, it is in general impossible to define the behavior of a network solely in static terms, since there may arise algebraic inconsistencies. This is illustrated by this simple network:



with for instance $\sigma(x) = x$; for which the output y is undefined if u is a nonzero input. One must explicitly introduce a processing delay and understand the behavior as an iteration, starting from some initial state. In graphical terms, a way to represent this situation is by adding to the definition of networks, besides neurons as described earlier, also dynamic elements, namely unit delays.



For instance, in this example, if the each initial value of y is y_0, then after one step one has y equal to $\sigma(y_0+u(0))$, where $u(0)$ is the input at "time 0", after two steps one has $\sigma(\sigma(y_0+u(0))+u(1))$, and so forth. In this manner, the behavior of a network is described by a system of difference equations — one equation for the state of each neuron — which are forced by the external inputs. One obtains discrete-time systems in the sense standard in system theory (see e.g. [5]).

An alternative way to model feedback nets is to use a continuous time scale and, accordingly, systems of differential instead of difference equations. This alternative corresponds to thinking of neurons as capacitors. It can be represented by using integrators as dynamic elements, and gives rise to continuous-time systems in the sense of system theory. As a simple example, consider the right part of Figure 1. The behavior of this net is described by the following system of differential equations:

$$\dot{x}_1(t) = \sigma_1(2x_1(t) + x_2(t) - u_1(t) + u_2(t)) \quad (3a)$$

$$\dot{x}_2(t) = \sigma_2(-x_2(t) + 3u_2(t)) \quad (3b)$$

$$y(t) = x_1(t), \quad (3c)$$

where the dot indicates derivative with respect to time, σ_1 and σ_2 are two activation functions, the input to the system is given by the vector $u(t) = (u_1(t), u_2(t))$, and the output at time t is the current value of x_1. (We usually omit arguments "t".)

One speaks of (discrete or continuous time) *feedback*, *recurrent*, or *dynamic* nets when dynamic elements are included.

One way to associate an input/output behavior to a dynamic net, given a specified set of initial conditions, is by looking at the last output that results after a finite-length input has been presented. For instance, in the above example, suppose that we fixed the initial state as $x_1(0) = x_2(0) = 0$. Then, given any input $u(t) = (u_1(t), u_2(t))$ defined for $t \in [0,T]$, we may solve the system of differential equations (3) with $u(\cdot)$ substituted into the right-hand side. (More precisely, one should assume, for instance, that σ satisfies a globally Lipschitz continuity condition, so that solutions are indeed defined for all $t \in [0,T]$, and that the input components are in $\mathcal{L}^\infty(0,\infty)$.) In this way we obtain a state trajectory $(x_1(t), x_2(t))$. We may then read-out the output $y(T) = x_1(T)$ and interpret it as the output of the network in response to the forcing function u.

2.1 Definitions: Feedforward Nets

By an (m,n)-*layer* we mean a triple

$$(A, a, \sigma) \quad (4)$$

consisting of a matrix $A \in \mathbb{R}^{n\times m}$, a vector $a \in \mathbb{R}^n$, and a diagonal mapping

$$\boldsymbol{\sigma} : \mathbb{R}^n \to \mathbb{R}^n : \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \mapsto \begin{pmatrix} \sigma_1(x_1) \\ \vdots \\ \sigma_n(x_n) \end{pmatrix}, \tag{5}$$

where $\sigma_1, \dots, \sigma_n$ are maps $\mathbb{R} \to \mathbb{R}$. The *mapping induced by the layer* $\Sigma = (A, a, \boldsymbol{\sigma})$ is the function

$$\varphi_\Sigma : \mathbb{R}^m \to \mathbb{R}^n : u \mapsto \boldsymbol{\sigma}(Au + a) \tag{6}$$

The component maps $\sigma_1, \dots, \sigma_n$ are the *activations of the layer*. If it is the case that all the σ_i are equal to a fixed function σ, we say that the layer is *homogeneous* with activation σ and write also $\vec{\sigma}^{(n)}$ instead of $\boldsymbol{\sigma}$.

Let k be a positive integer. A *k-hidden layer net* is a $k+1$-tuple $(\Sigma^1, \dots, \Sigma^{k+1})$ of layers, where Σ^{k+1} is homogeneous with identity activation $\sigma(x) = x$. The *mapping induced by the net* $(\Sigma^1, \dots, \Sigma^{k+1})$, where each Σ^i is an (n_{i-1}, n_i)-layer, is the composition

$$\varphi_{\Sigma^{k+1}} \circ \varphi_{\Sigma^k} \circ \cdots \circ \varphi_{\Sigma^1} : \mathbb{R}^m \to \mathbb{R}^p$$

where $m = n_0$ and $p = n_{k+1}$. The integer $n := n_1 + \ldots + n_k$ is the *number of neurons* of the net. The spaces $\mathbb{R}^m$ and $\mathbb{R}^p$ are called, respectively, the input and output spaces of the net. (The layers $\Sigma^1, \dots, \Sigma^k$ are often called hidden layers to reflect the fact that the signals in the intermediate spaces are not part of inputs or outputs and in that sense they are "hidden" from the external environment.) The *activations of the net* are the activations of the layers $\Sigma^1, \dots, \Sigma^k$. We say that the net is *homogeneous* with activation σ if all the activations of the layers Σ^i, $i = 1, \dots, k$ are equal to the fixed function σ. Obviously, a vector mapping $f : \mathbb{R}^m \to \mathbb{R}^p$ is induced by some net with activations belonging to a given set S if and only if each of its components $f_i : \mathbb{R}^m \to \mathbb{R}$ is induced by some net with activations from this set S. Thus one often studies only scalar-output nets.

As an illustration, take again the diagram in the left part of Figure 1, ignoring both the right diagram and the dotted line for now. This is a pictorial representation of a 2-hidden layer net whose $(2,2)$-layer Σ^1 is defined by

$$A = \begin{pmatrix} 5 & -1 \\ 1 & 2 \end{pmatrix}, \quad a = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{\sigma} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}$$

while the $(2,2)$-layer Σ^2 is defined by

$$A = \begin{pmatrix} 3 & 2 \\ 0 & -3 \end{pmatrix}, \quad a = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \boldsymbol{\sigma} = \begin{pmatrix} \sigma_3 \\ \sigma_4 \end{pmatrix}$$

and the $(2,1)$-layer Σ^3 has $A = (2 \quad 5)$ and $a = 0$. The mapping induced by this net is

$$\mathbb{R}^2 \to \mathbb{R} : u \mapsto 2\sigma_4\Big(3\sigma_1(5u_1 - u_2) + 2\sigma_2(u_1 + 2u_2 + 1) + 1\Big) + 5\sigma_3\Big(-3\sigma_2(u_1 + 2u_2 + 1) - 1\Big).$$

2.1.1 Particular Case: Single-Hidden Layer Nets

A particular case, which appears often in the discussions to follow, is that of homogeneous, scalar-output, single hidden layer nets (just *1HL* from now on). A function f can be expressed as the input/output mapping induced by a 1HL net with activation σ if and only if f belongs to the affine span of the (multivariate) dilations and translates of σ:

$$f(u) \;=\; c_0 + \sum_{i=1}^{n} c_i\sigma(A_i u + a_i) \;=\; C\vec{\sigma}^{(n)}(Au + a) + c_0 \,, \tag{7}$$

where we have taken $\Sigma^1 = (A, a, \vec{\sigma}^{(n)})$, $\Sigma^2 = (C, c_0, \mathrm{id})$, and we are denoting by A_i the ith row of the matrix A, $a = \mathrm{col}(a_1, \ldots, a_n)$, and $C = (c_1, \ldots, c_n)$. Letting $A = (a_{ij})$ and $u = \mathrm{col}(u_1, \ldots, u_m)$, the corresponding 1HL net may be represented diagrammatically as in Figure 3.

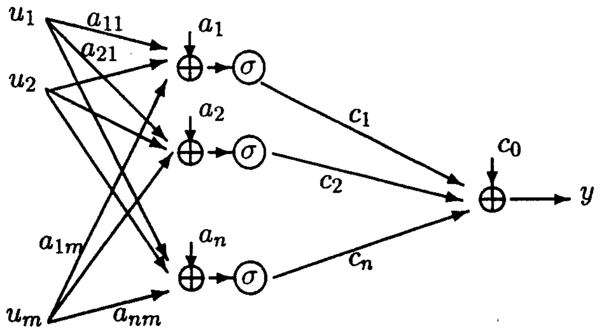


Figure 3: *Homogeneous Scalar-output One Hidden Layer (1HL) Net*

The interest in 1HL nets, besides their simplicity and their resemblance to related mathematical constructs (functional expansions of various types, wavelets, etc) is due in large part to the density properties, of functions computed by 1HL nets, with respect to various spaces (continuous functions, L^p spaces, $p < \infty$, Sobolev spaces under additional differentiability hypotheses). It is by now well-known, thanks to the pioneering work of Cybenko, Hornik, White, Leshno, and others. that 1HL networks have these universal approximation properties, assuming very little on the activation σ besides it not being polynomial (for simple proofs, which apply to restricted yet quite general activations, see [32]).

Of course. one should not read too much into this universality property, since better rates of convergence in function approximation, or better classifiers for pattern recognition, or better feedback control laws, may be obtainable with more complex networks than 1HL ones. (As an analogy, polynomials are dense in spaces of continuous functions but they are not necessarily the right choice in approximation problems, where other families of functions, such as splines, are often preferred.) This has been remarked often in the literature; see for instance [10].

In fact, two-hidden layer nets turn out to be more appropriate, from a theoretical standpoint. for certain problems. For instance, the characteristic function of a square in the plane can be easily approximated by the output of a homogeneous two-hidden layer net (with various types of σ). but good approximations using 1HL nets require many terms (no matter what σ). A basic problem is that uniform approximation of discontinuous functions is in general impossible using 1HL nets; technically, there is no density on L^∞. even on compact subsets; as a matter of fact, an even stronger negative result holds regarding the impossibility of constructing sections of certain coverings, as discussed in [10]. This has serious implications in solving inverse problems, such as inverse kinematics computations in robotics, and indeed has been noted experimentally by many authors. The same problem appears in control applications, as explained later.

2.2 Definitions: Feedback Nets

By an n-dimensional, m-input, p-output *recurrent net* we mean a quadruple $(A, B, C, \boldsymbol{\sigma})$ consisting of three matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ and a diagonal mapping $\boldsymbol{\sigma}$ as in Equation (5). An *initialized recurrent net* is a 5-tuple

$$\Sigma = (A, B, C, x^0, \boldsymbol{\sigma}) \tag{8}$$

consisting of a recurrent net $(A, B, C, \boldsymbol{\sigma})$ together with a vector $x^0 \in \mathbb{R}^n$. The (discrete or continuous time) *system induced by the net* (8) is the set of n coupled (difference or differential, respectively) equations, plus measurement function:

$$x(t+1)\,[\text{or}\,\dot{x}(t)\,] = \boldsymbol{\sigma}\,(Ax(t) + Bu(t))\,,\ \ x(0) = x^0\,,\quad y(t) = Cx(t)\,. \tag{9}$$

The component maps $\sigma_1, \ldots, \sigma_n$ of $\boldsymbol{\sigma}$ are the *activations of the net*. If it is the case that all the σ_i are equal to a fixed function σ, we say that the net is *homogeneous* with activation σ and write also $\vec{\sigma}^{(n)}$ instead of $\boldsymbol{\sigma}$. The spaces $\mathbb{R}^m$, $\mathbb{R}^n$, and $\mathbb{R}^p$ are called respectively the input, state, and output spaces of the net.

As an illustration, take again the diagram in the right part of Figure 1. This is a pictorial representation of the 2-dimensional, 2-input, single-output recurrent net defined by

$$A = \begin{pmatrix} 2 & 1 \\ 0 & -1 \end{pmatrix}\,,\ B = \begin{pmatrix} -1 & 1 \\ 0 & 3 \end{pmatrix}\,,\ C = (1 \quad 0)\,,\ \boldsymbol{\sigma} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}$$

and inducing the system (3).

In the present context, one interprets the vector equations for x in (9) as representing the evolution of an ensemble of n "neurons," where each coordinate x_i of x is a real-valued variable which represents the internal state of the ith neuron, and each coordinate $u_i, i = 1, \ldots, m$ of u is an external input signal. The coefficients A_{ij}, B_{ij} denote the weights, intensities, or "synaptic strengths," of the various connections. The coordinates of $y(t)$ represent the output of p probes, or measurement devices, each of which averages the activation values of many neurons. Often C is just a projection on some coordinates, that is, the components of y are simply a subset of the components of x. Several motivations for the study of recurrent nets are mentioned in sections 1.3 and 2.4.

The linear systems customarily studied in control theory are precisely the homogeneous recurrent nets with identity activation and $x^0 = 0$. This formal analogy to linear systems, for which a very detailed theory has been developed (see e.g. the textbook [5]) turns out to be very fruitful, as we shall explain later.

One also writes (9) simply as $\Delta x = \boldsymbol{\sigma}\,(Ax + Bu)$, $x(0) = x^0$, $y = Cx$, where $\Delta x = x^+$ (time-shift) or $\Delta x = \dot{x}$ (time derivative) in discrete or continuous time respectively. Figure 4 gives a "block diagram" of (9).

To each initialized recurrent net $(A, B, C, x^0, \boldsymbol{\sigma})$ we associate a discrete time and a continuous time input/output behavior. Assume given a sequence $u = u(0), \ldots, u(k-1)$ of elements of the input space $\mathbb{R}^m$. One may iteratively solve the difference equation (9) starting with $x(0) = \xi$, thereby obtaining a sequence of state vectors $x(1), \ldots, x(k)$. In this manner, each initialized recurrent net induces a mapping, on inputs of fixed length k,

$$\lambda_\Sigma^k \,:\, (\mathbb{R}^m)^k \to \mathbb{R}^p \,:\, u \mapsto y(k) = Cx(k) \tag{10}$$

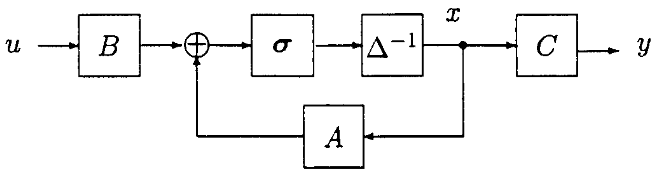


Figure 4: *Recurrent Net*

which assigns to the input u the last output produced in response. Sometimes one also introduces $\lambda_\Sigma : \bigcup_k (\mathbb{R}^m)^k \to \mathbb{R}^p$ on all possible sequences by letting the restriction of λ_Σ to $(\mathbb{R}^m)^k$ be λ_Σ^k. In continuous time, analogous maps are defined on inputs $u : [0,k] \to \mathbb{R}^m$, through solution of an initial value problem with $x(0) = \xi$. This requires certain technical hypotheses (for instance, Lipschitz continuous activations, and inputs that are measurable and locally integrable) which we do not need to discuss here; see [5] for the precise definitions.

Several variations of the above model can be found in the literature, such as systems of the following form:

$$\dot{x} \; = \; -Dx + \boldsymbol{\sigma}(Ax + Bu)\,, \quad y = Cx\,,$$

where D a diagonal matrix (for "Hopfield nets" one picks in addition A symmetric) and A, B, C are as before. Observe that, if D has negative entries, and if the activations are bounded, all solutions $x(t)$ in this equation are bounded (because the linear term dominates, for large x); it is this stability property that makes the variation appealing in applications. In other variants, the input term Bu may be outside the nonlinearity. The paper [14] showed how, at least for certain problems, it is possible to transform among the different models, in such a way that once that results are obtained for (9), corollaries for the variants are easily obtained. Thus, here we restrict attention to the simpler form (9), which has the advantage of requiring one less object (no need for the matrix D) for its specification.

One may broaden the notion of initialized recurrent net by allowing "biases" or "offsets", i.e. nonzero vectors $d \in \mathbb{R}^n$ and $e \in \mathbb{R}^p$ in the update and the measurement equations respectively. These equations would then take the more general form $x^+ = \boldsymbol{\sigma}(Ax+Bu+d)$, $y = Cx+e$. Despite the fact that biases are useful, we do not need to include such an extension in the formal definition. This is because the input/output behavior of any such net also arises as the input/output behavior of a net in the sense defined earlier (zero biases), with state space $\mathbb{R}^{n+1}$ and same activations. The simulation is achieved by means of the introduction of an additional variable z whose value is constantly equal to a nonzero number z_0 in the range of one of the activations, say σ, in such a manner that the equations become $x^+ = \boldsymbol{\sigma}(Ax + zd' + Bu)$, $z^+ = \sigma(a_0 z)$, $y = Cx + ze'$, where a_0 is chosen so that $\sigma(a_0 z_0) = z_0$ and d', e' are so that $z_0 d' = d$ and $z_0 e' = e$ (if the only activation is $\sigma \equiv 0$, there would be nothing to prove).

2.3 Architectures

Roughly, by an "architecture" one means a choice of interconnection structure and of the activation functions σ for each neuron, leaving weights and initial states unspecified, as parameters. One may also stipulate that the initial state, or just certain specific coordinates of it, should be zero (as with linear systems in control theory). Once than the architecture is fixed, feedforward neural networks provide parametric families of functions, alternatives to more traditional parametric sets of functions such as polynomials, splines, rational functions, finite Fourier expansions, or wavelets, and of potential use in areas such as function approximation and interpolation. Similarly, feedback networks with a fixed architecture provide parametric classes of dynamical systems, and as such, universal models for identification and adaptive control. We formalize the notion of architecture by means of incidence matrices, employing binary matrices in order to specify the allowed interconnection patterns and initial states.

By an (m,n)-*layer architecture* we mean a triple

$$(I, J, \sigma) \tag{11}$$

consisting of a matrix $I \in \{0,1\}^{n\times m}$, a vector $J \in \{0,1\}^n$, and a diagonal mapping σ as in Equation (5). As before, we call the component maps $\sigma_1, \dots, \sigma_n$ the *activations of the layer architecture*. Let k be a positive integer. A *k-hidden layer net architecture* is a $k+1$-tuple $\mathcal{A} = (\mathcal{L}^1, \dots, \mathcal{L}^{k+1})$ of layer architectures, where $\mathcal{L}^{k+1}$ is homogeneous with identity activation $\sigma(x) = x$. As earlier, we say that the architecture is homogeneous with activation σ if all the activations of the layers $\mathcal{L}^i$, $i = 1, \dots, k$ are equal to the fixed function σ. A *net with architecture* $\mathcal{A}$ is an instantiation obtained by choosing values for the nonzero entries, that, is, any k-hidden layer net $(\Sigma^1, \dots, \Sigma^{k+1})$ which is so that, for every $i = 1, \dots, k+1$ the following property holds: if $\Sigma^i = (A, a, \sigma)$ and $\mathcal{L}^i = (I, J, \sigma')$, then $\sigma = \sigma'$ and the entries of the matrix and vector satisfy $A_{\mu\nu} = 0$ whenever $I_{\mu\nu} = 0$ and $a_\mu = 0$ whenever $J_\mu = 0$.

Let $\mathcal{A} = (\mathcal{L}^1, \dots, \mathcal{L}^{k+1})$, where $\mathcal{L}^i$ is an (n_{i-1}, n_i)-layer (I^i, J^i, σ^i). We again call the integer $n := n_1 + \dots + n_k$ the number of neurons. Suppose that the binary matrix I^i has exactly λ_i nonzero entries and the binary vector J^i has exactly μ_i nonzero entries. Then we say that the number $r := \lambda_1 + \dots + \lambda_{k+1} + \mu_1 + \dots + \mu_{k+1}$ is the number of *parameters* or *weights* of $\mathcal{A}$, and call $\mathbb{R}^r$ the parameter or weight space, and, as before, $\mathbb{R}^m$ and $\mathbb{R}^p$ the input and output spaces of $\mathcal{A}$. Order the indices of the nonzero entries of the I^i's and J^i's in any fixed manner, for instance by listing the nonzero entries of I^1 row by row, then those of J^1, and so on up to J^{k+1}. These indices are in one-to-one correspondence with the coordinates of vectors in $\mathbb{R}^r$. In this manner, one may view the architecture $\mathcal{A}$ as inducing a mapping

$$f_{\mathcal{A}} : \mathbb{R}^r \times \mathbb{R}^m \to \mathbb{R}^p$$

where $f(\rho, \cdot)$ is the mapping induced by the net $(\Sigma^1, \dots, \Sigma^{k+1})$ that would result if we substituted the parameters ρ into the nonzero entries. We denote by

$$\mathcal{F}_{\mathcal{A}} := \{f_{\mathcal{A}}(\rho, \cdot), \rho \in \mathbb{R}^r\} \tag{12}$$

the class of functions $\mathbb{R}^m \to \mathbb{R}^p$ thus associated to the architecture $\mathcal{A}$.

As an example, the diagram in the left part of Figure 1 is a net with architecture $(\mathcal{L}^1, \mathcal{L}^2, \mathcal{L}^3)$, where $\mathcal{L}^1$ has

$$I = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad J = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \sigma = \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}$$

(we used a dotted arrow in Figure 1 to exhibit a bias that is necessarily zero, due to the zero entry in J), $\mathcal{L}^2$ has

$$I = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad J = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \sigma = \begin{pmatrix} \sigma_3 \\ \sigma_4 \end{pmatrix},$$

and layer $\mathcal{L}^3$ has $I = (1 \quad 1)$ and $J = 0$. The mapping induced by this architecture is

$$f_{\mathcal{A}} : \mathbb{R}^{12} \times \mathbb{R}^2 \to \mathbb{R}$$

where $f(\rho,u)$ equals:

$$\rho_{11}\sigma_4\Big(\rho_6\sigma_1(\rho_1u_1+\rho_2u_2)+\rho_7\sigma_2(\rho_3u_1+\rho_4u_2+\rho_5)+\rho_9\Big)+\rho_{12}\sigma_3\Big(\rho_8\sigma_2(\rho_3u_1+\rho_4u_2+\rho_5)+\rho_{10}\Big)$$

and $\mathcal{F}_{\mathcal{A}}$ is the set of functions $f(\rho,\cdot)$ obtained for each fixed choice of the parameter vector ρ.

We next define similar notions for the feedback case. By an n-dimensional, m-input, p-output *recurrent architecture* we mean a 5-tuple

$$\mathcal{A}=(\alpha,\beta,\gamma,\xi,\boldsymbol{\sigma}) \tag{13}$$

consisting of three matrices $\alpha\in\{0,1\}^{n\times n}$, $\beta\in\{0,1\}^{n\times m}$, and $\gamma\in\{0,1\}^{p\times n}$, a vector $\xi\in\{0,1\}^n$, and a diagonal mapping $\boldsymbol{\sigma}$ as in Equation (5).

An *initialized recurrent net with architecture* $\mathcal{A}$ is an instantiation obtained by choosing values for the nonzero entries, that, is, any initialized recurrent net $(A,B,C,x^0,\boldsymbol{\sigma}')$ such that $\boldsymbol{\sigma}=\boldsymbol{\sigma}'$ and the entries of the matrices and vector satisfy $A_{ij}=0$ whenever $\alpha_{ij}=0$, $B_{ij}=0$ whenever $\beta_{ij}=0$, $C_{ij}=0$ whenever $\gamma_{ij}=0$, and $x_i^0=0$ whenever $\xi_i=0$.

We say also here that the component maps $\sigma_1,\ldots,\sigma_n$ of $\boldsymbol{\sigma}$ are the activations of the net, which is homogeneous with activation σ if all σ_i are equal to a fixed function σ. The spaces $\mathbb{R}^m$, $\mathbb{R}^n$, and $\mathbb{R}^p$ are respectively the input, state, and output spaces of the architecture. Suppose that the binary matrices α, β, and γ and the vector ξ have exactly κ, λ, μ, and ν nonzero entries respectively; then we call the number $r:=\kappa+\lambda+\mu+\nu$ the number of *parameters* or *weights* of $\mathcal{A}$, and call $\mathbb{R}^r$ the parameter or weight space. Arrange the indices of the nonzero entries in any fixed manner, for instance by listing their nonzero entries row by row, for α, β, γ, and ξ in that order. These indices are in one-to-one correspondence with the coordinates of vectors in $\mathbb{R}^r$. In this manner, one may view the architecture $\mathcal{A}$ as representing a parameterized system (in continuous or discrete time) $\Delta x=\boldsymbol{\sigma}(\alpha x+\beta u)$, $x(0)=\xi$, $y=\gamma x$ where, by substituting the parameters $\rho\in\mathbb{R}^r$ into the nonzero entries of $(\alpha,\beta,\gamma,\xi)$, every possible initialized recurrent net $\Sigma=\mathcal{A}(\rho)$ with architecture $\mathcal{A}$ results.

For example, the diagram in the right part of Figure 1 is a recurrent net with architecture $\mathcal{A}$, where

$$\alpha=\begin{pmatrix}1&1\\0&1\end{pmatrix},\quad \beta=\begin{pmatrix}1&1\\0&1\end{pmatrix},\quad \gamma=(1\quad 0),\quad \boldsymbol{\sigma}=\begin{pmatrix}\sigma_1\\\sigma_2\end{pmatrix}$$

and the (continuous-time) corresponding parameterized system (with parameter space $\mathbb{R}^7$) is

$$\dot{x}_1(t) = \sigma_1(\rho_1x_1(t)+\rho_2x_2(t)+\rho_4\boldsymbol{u}_1(t)+\rho_5u_2(t)) \tag{14a}$$
$$\dot{x}_2(t) = \sigma_2(\rho_3x_2(t)+\rho_6u_2(t)) \tag{14b}$$
$$y(t) = \rho_7x_1(t)\,. \tag{14c}$$

Recalling the notations in Equation (10), for each recurrent architecture $\mathcal{A}$ and each $k>0$, we may introduce the set

$$\mathcal{F}_{\mathcal{A},k}:=\left\{\lambda_\Sigma^k\,,\ \Sigma=\mathcal{A}(\rho),\,\rho\in\mathbb{R}^r\right\} \tag{15}$$

of mappings $(\mathbb{R}^m)^k\to\mathbb{R}^p$. Elements of this set are the input/output mappings induced on inputs of length k by each possible initialized recurrent net with architecture $\mathcal{A}$.

2.4 A General Framework for Learning

In this section. we briefly review a formulation of the problem of learning. Our purpose in doing so is to motivate, and to set the stage for a description of, some of our main lines of research. We chose to base the discussion in terms of the paradigm currently popular in computational learning theory (COLT), a formulation which also appears, in a slightly different form, in related statistics areas of parametric and nonparametric estimation. (The COLT area, considered part of theoretical computer science, is one of considerable activity at present. as evidenced by the profusion of papers and conferences on the subject.) The language to be introduced affords a way to systematically explore the capabilities of neural networks in the context of a precise mathematical formalism, but alternative formulations of the same fundamental questions are also possible, based on other methods from statistics and numerical analysis.

The questions that we deal with can informally be described as concerning the choice of suitably "good" responses (outputs) to given stimuli (inputs). In applications. this may translate into tasks as diverse as: in state feedback control problems, finding a control that produces a desired action as a function of the current state; in OCR algorithms. guessing the correct character given a digitized image of a letter; or in market analysis programs, deciding whether to buy or sell a stock based on a time-window of price data. "Learning" comes into the picture if we assume that, as part of a preliminary "training" phase, we were supplied with a representative sample of such "good" responses, and our goal is to form associations which allow us to produce good responses for future inputs, including of course inputs which were not seen exactly during the training period (we haven't seen all possible apples and oranges, yet can reliable distinguish between them). It is this last quality of generalization (interpolation/extrapolation in an abstract sense) that makes the problem interesting: otherwise we could just in principle simply store all training data, and use efficient database retrieval methods for finding appropriate responses. One could say that, in some implicit sense, the goal of a learning system is to extract decision rules from training data, although the formalization to be discussed next does not make rule-extraction explicit.

Of course, the problem of generalization is not well-posed unless we make some prior assumptions about what the "good" input/output pairs are. In numerical analysis, such prior assumptions take the form of smoothness constraints; in classical statistics one postulates e.g. normality of distributions. The general approach in learning theory is to postulate that there is a probability distribution P on input/output pairs (u, y) which generates the "good" pairs, in the sense that the desirable responses for a given u are those y for which the probability of y given the observed u is in some sense maximized. The distribution P may be unknown, but it will be assumed to be a member of a fixed class of distributions $\mathcal{P}$ (e.g., normal distributions with different means and standard deviations. uniform distributions with different supports, or even the set of all probability measures with respect to a fixed σ-algebra). The most important assumption is one of stationarity: both the training samples and the data seen in the future (testing data) are assumed to be randomly drawn according to the same P.

This general idea includes many cases of interest. A most important special case is that of *function learning*, in which the "good" pairs are those that belong to the graph of an unknown "target" (or "oracle," or "teacher") function h, whose behavior the learner attempts to emulate. We assume that inputs u are generated randomly according to a probability distribution P_0, and for each input so generated, the target value $y = h(u)$ is provided to the learning system. (At this level of abstraction, function learning includes systems identification, in which case h would represent a plant to be identified from its responses $h(u)$ to, for instance, white noise inputs u.) This fits in the above setting roughly as follows. There is a natural probability P_h induced on pairs (u, y): (u, y) has the same probability as $(u, h(u))$, and has zero probability otherwise. Estimating P_h amounts to learning h. A related example is that in which the learner only has access to noisy measurements of $h(u)$; in that case, one may in an obvious manner induce a probability on pairs (u, y) which incorporates the effects of noise. (Of course, we are being extremely informal right here, for instance because what we just said is not correct for continuous domains, where one must argue in terms of probabilities with respect to suitable σ-algebras: the purpose is to explain intuitively the basic definitions. Much more discussion can be found in standard texts in machine learning and learning theory, including explanations of relations to Bayesian analysis and classical statistical techniques.)

In evaluating performance of a learning system, one has to allow for the facts that, (1) accidentally, training data may not have been rich enough, as a sample, to identify the unknown distribution, which means that our confidence in future predictive behavior can never be perfect, and (2) even if the training data was indeed representative, that it may well be the case that a new, unseen, input to be responded to may itself be an "outlier" with respect to typical inputs, so our prediction may not be totally accurate. These two potential sources of uncertainty give rise to the notion of "probably approximately correct" learning, in which the goal is to be able to provide, with high confidence, a reasonably accurate response to future inputs. (Two numbers in the range $[0, 1]$, typically denoted by ε and δ, appear in all formulations, to denote respectively these accuracy and confidence levels.)

2.4.1 PAC Learning

We now review briefly the basic "agnostic PAC learning" formalism. Two sets $\mathcal{U}$ and $\mathcal{Y}$ are given, the sets of ***input*** and ***output*** values respectively. Although the theory can be derived in more generality, we will assume here that $\mathcal{U}$ is a complete separable metric space (typically, in any case, $\mathcal{U}$ is a closed subset of an Euclidean space) and that $\mathcal{Y}$ is a compact subset of $\mathbb{R}$ (far more general outputs could equally well be considered, but notations become slightly more involved). Also given is a family of Borel probability measures $\mathcal{P}$ on $\mathcal{U} \times \mathcal{Y}$ and a family $\mathcal{F}$ of Borel measurable maps $f : \mathcal{U} \to \mathcal{Y}$.

By an *identifier* we mean a map $\varphi : \Omega \to \mathcal{F}$, where we are denoting by Ω the set of all finite sequences $\omega = (u_1, y_1), \ldots, (u_s, y_s)$ of elements of $\mathcal{U} \times \mathcal{Y}$ (with varying lengths s). We write the value of φ on a sequence ω as φ_ω; thus φ_ω is itself a function $\mathcal{U} \to \mathcal{Y}$.

As discussed in previous paragraphs, the role of a $P \in \mathcal{P}$ is to represent the distribution of input/output data to be learned, while an element $f \in \mathcal{F}$ describes the function used by the learner to fit the data. In our discussions, $\mathcal{F}$ will usually be the class $\mathcal{F}_\mathcal{A}$ (cf. Equation(12)) consisting of those mappings $f(\rho, \cdot)$ associated to the possible nets with a given architecture $\mathcal{A}$. The elements of $\mathcal{F}$ are often called *hypotheses*. Thus an identifier is a method of assigning a candidate hypothesis $f \in \mathcal{F}$ based on the training data ω that has been observed in the past (we prefer not to use the term "algorithm" for φ, since this term connotes computability, and we wish to separate questions of computability from information-theoretic concepts). We next quantify the performance of identifiers, assuming that the training and test data is being drawn according to a distribution $P \in \mathcal{P}$.

Assume that a probability $P \in \mathcal{P}$ has been chosen. The *error* of the identifier φ with respect to the probability measure P and the (training) sequence $\nu \in \Omega$ is defined as the expectation (with respect to P) of the squared error ("risk"), on test data, of the function produced by the identifier:

$$\mathrm{Err}\,(P,\varphi,\nu) \;:=\; \int_{\mathcal{U}\times\mathcal{Y}} (y-\varphi_\nu(u))^2\,dP(u,y)\,.$$

(We use squared-error for simplicity of exposition, but more arbitrary loss criteria could be employed as a definition.) In a special case of wide interest, that of binary classification problems, the output value set $\mathcal{Y}$ consists just of two elements $\{0,1\}$; in that case, the expression $\mathrm{Err}\,(P,\varphi,\nu)$ simply represents the probability $y \neq \varphi_\nu(u)$ of misclassification.

The best achievable performance (lowest possible error) among *all* $f \in \mathcal{F}$, if the test samples are distributed according to P, is:

$$\underline{\mathrm{Err}}\,(P) \;:=\; \inf_{f\in\mathcal{F}} \int_{\mathcal{U}\times\mathcal{Y}} (y-f(u))^2\,dP(u,y)\,.$$

Since $\varphi_\nu \in \mathcal{F}$, $\underline{\mathrm{Err}}\,(P) \leq \mathrm{Err}\,(P,\varphi,\nu)$, for all identifiers φ and all training data ν. A measure of merit for any given φ can be formulated in terms of the gap between these numbers: we ask that, for most training instances, the gap be small. More precisely, an identifier φ is said to be *probably approximately correct (PAC)* with respect to the family of probabilities $\mathcal{P}$ and the hypothesis class $\mathcal{F}$, if for each $\varepsilon > 0$ and each $\delta > 0$ there is some integer $s = s(\varepsilon,\delta)$ such that, for every $P \in \mathcal{P}$,

$$P^s\Big\{\mathrm{Err}\,(P,\varphi,\nu) < \underline{\mathrm{Err}}\,(P) + \varepsilon\Big\} \;>\; 1-\delta\,. \qquad (16)$$

Note that the probability on training samples ν is being understood with respect to the induced measure P^s on the product space $(\mathcal{U}\times\mathcal{Y})^s$. In other words, one is asking that the event "$|\mathrm{Err}\,(P,\varphi,\nu) - \underline{\mathrm{Err}}\,(P)| < \varepsilon$" becomes almost sure as $s \to \infty$, uniformly on $P \in \mathcal{P}$. If φ is PAC, we may define a function $s : \mathbb{R}_{>0} \times \mathbb{R}_{>0} \to \mathbb{Z}$ by letting $s(\varepsilon,\delta)$ be the smallest integer for which equation (16) holds for all $P \in \mathcal{P}$; this function is often called the *sample complexity* of φ.

If a PAC identifier exists, we say that $(\mathcal{P},\mathcal{F})$ is PAC (or *uniformly*) *learnable*. The most desirable special case is that in which $\mathcal{P}$ is the family of *all* (Borel) probability measures; if $(\mathcal{P},\mathcal{F})$ is PAC learnable with this $\mathcal{P}$, one says simply that $\mathcal{F}$ itself is PAC learnable.

From the work of Vapnik, Pollard, Haussler, and others, there follows a simple yet powerful sufficient combinatorial condition for PAC learnability of $\mathcal{F}$: *if $\mathcal{F}$ has finite pseudo-dimension, then it is PAC learnable.* (We review in Section 3.2 the meaning of pseudo-dimension, and in particular the notion of Vapnik-Chervonenkis dimension. They characterize the richness of $\mathcal{F}$ as a class of functions, expressed in terms of discrimination power.) Furthermore, in that case it is possible, at least in principle, to design an identifier using the most "naive" approach of fitting an $f \in \mathcal{F}$ as well as possible to the observed data, and then using this f for prediction. We make this precise next.

Given a sample $\nu = (u_1,y_1),\ldots,(u_s,y_s)$, for each possible function $f \in \mathcal{F}$ we may consider the quantity

$$\mathrm{Emp}\,(f,\nu) \;:=\; \sum_{i=1}^{s}\Big(f(u_i)-y_i\Big)^2$$

which represents the "fitting error" or "empirical risk" that f makes on the given sample. Consider also, for this same sample, the smallest possible error achievable using the given hypotheses class:

$$\underline{\mathrm{Emp}}\,(\nu) \;:=\; \inf_{f\in\mathcal{F}} \mathrm{Emp}\,(f,\nu)\,. \qquad (17)$$

Note that the calculation of $\underline{\mathrm{Emp}}\,(\nu)$, for observed training data ν, as well as the finding of an element $f \in \mathcal{F}$ which provides a near optimal value (i.e., so that $\mathrm{Emp}\,(f,\nu) - \underline{\mathrm{Emp}}\,(\nu)$ is small), constitutes an optimization problem over $\mathcal{F}$. When $\mathcal{F}$ is specified in terms of a a finite number of parameters (such as the adjustable weights in a neural network), this becomes a problem of finite-dimensional optimization. With these notations we can state the by-now classical results:

If the pseudo-dimension d of $\mathcal{F}$ is finite, then:

1. *there is an identifier with sample complexity*

$$s(\varepsilon,\delta) \;\leq\; \frac{c}{\varepsilon^2}\left(d\ln\frac{1}{\varepsilon} \;+\; \ln\frac{1}{\delta}\right).$$

where c is a constant[*] that depends on $\mathcal{Y}$, and

2. *such an identifier can be obtained by **empirical risk minimization**: to achieve the error estimate in Equation (16), for a given δ, ε, and given the sample $\nu = (u_1, y_1), \ldots. (u_s, y_s)$ with length $s \geq s(\varepsilon, \delta)$, it is enough to find **any** function $f \in \mathcal{F}$ with the following property:*

$$\mathrm{Emp}\,(f,\nu) < \underline{\mathrm{Emp}}\,(\nu) + \frac{\varepsilon}{3}\,. \tag{18}$$

and to define φ_ν to be this function f.

(Even more interesting, this almost-minimization may be performed by a probabilistic algorithm, in that it is enough that the almost-optimality estimate in (18) hold with high probability.) Through empirical risk minimization, the theory of PAC learning makes contact with, and provides an elegant generalization of, classical questions regarding the uniform convergence of empirical means of random variables and. more generally, probabilities of sets, and in particular the Glivenko-Cantelli Lemma.

It should be emphasized that the finiteness of pseudo-dimension is merely a sufficient, not a necessary, condition, and that much research nowadays concerns the development of weaker conditions for learnability (viz. the area of "fat shattering"). However, as we shall see. it is a very useful condition. and, moreover, in the special case of binary valued outputs, which is the case of interest in pattern recognition and classification, the condition turns out to be necessary as well.

2.4.2 Subproblems

The above discussion leads to the following general questions, for any given $\mathcal{F}$ and $\mathcal{P}$:

Q1 How large can the errors $\underline{\mathrm{Err}}\,(P)$ be?

Q2 Does $\mathcal{F}$ have finite pseudo-dimension?

Q3 Is it computationally feasible to find $\underline{\mathrm{Emp}}\,(\nu)$?

Q4 What are the properties of the error function being minimized in (17)?

The first question characterizes the minimum potentially achievable error, no matter how many samples are seen or how powerful an algorithm is used: the smaller $\underline{\mathrm{Err}}\,(P)$, the more useful is the estimate in Equation (16). It leads to function approximation (of densities, if the

[*] A far more explicit estimate is known, but we absorb many constants into c in order to exhibit the dependence on d, ε, δ as simply as possible; we also must assume that ε is small enough, e.g. $\varepsilon < 1/2$.

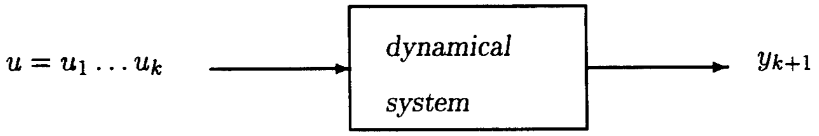
P's are described in such terms) by elements of $\mathcal{F}$. The line of work in the PI papers [44] and [21] was motivated in this manner; see Section 3.3.1 of the report. The second question. whose answer helps determine if it is possible to learn at all (using finite samples) is more combinatorial in nature. and in particular the study of lower bounds relies on finding explicit constructions to implement a large variety of functions. In this direction. relevant PI work is represented by [8], [11]. [39]. [53], [25], and [26] as well as related results for recurrent nets to be mentioned later; see Sections 3.2.1 and 3.2.3 of the report. The third question is of a computational complexity nature (NP-completeness), and its study, at least for certain activations, requires simple techniques from combinatorics and discrete mathematics. Relevant PI work is exemplified by [48] and [20]; see Section 3.5 of the report. Finally, the last question, whose answer impacts the performance of numerical algorithms based on gradient descent (so-called "backpropagation"), leads to issues of Morse theory, stratifications of subanalytic sets, model theory on logic, as well as elementary analysis. Among PI work in this direction are the references [7], [9], [50], [22]; see Section 3.1 of the report.

Section 3 of the report discusses some of these issues in more detail, though space constraints do not allow a technical development and the references should be consulted.

2.4.3 Learning and Recurrent Nets

As explained in the introduction to this report, in many applications involving pattern classification or learning, input data arrives naturally as a *time series*. Inputs fed into a speech recognition system are often sequences consisting of windowed Fourier coefficients and in control problems inputs to a regulator may be sequences of measurements of the plant being controlled as well as the successive coordinates of a path to be tracked. Under these circumstances, a learning system should exploit the information inherent in the correlations and dependencies that exist among the terms of the input sequence. One way to take into account this additional structure is through the use of hypotheses classes which consist of dynamical systems. Kalman filtering, which relies on linear dynamical systems for extracting information (filtering of noise) from a stream of data, is perhaps the most successful known example of an application of this principle.

In the notations and terminology established earlier, and for concreteness dealing with the discrete-time case, the inputs u, elements of the set $\mathcal{U}$, are now finite sequences (which. depending on the problem being studied, could be of a fixed length, or of varying lengths). We assume that the output space is $\mathbb{R}$. The hypotheses classes $\mathcal{F}$ consist of systems which start from a fixed initial state. evolve according to internal dynamics forced by the external input sequence $u = u_1, \ldots, u_k$, and produce real-number outputs $y_t, t = 1, \ldots$ as a result. In a learning context, we will always view the *last* output, produced after presentation of the complete sequence u, as the output value $f(u)$ assigned by the hypothesis $f \in \mathcal{F}$ to the input sequence u.



Intuitively, by limiting the memory and power (dynamic order, number of adjustable parameters) of the elements of $\mathcal{F}$, and analyzing behavior for longer and longer input sequences, one is able to focus on the properties that truly reflect the dependence of $f(u)$ on long-term time correlations in the data. Although this paradigm is inspired by the use of finite automata for the recognition of languages, or the use of recursive least squares techniques in statistical problems. the interest here is in nonlinear, continuous-state, dynamical systems. In particular, we employ (scalar-output) recurrent networks and study classes such as $\mathcal{F}_{\mathcal{A},k}$, introduced in Equation 15, which consist of those mappings $(\mathbb{R}^m)^k \to \mathbb{R}$ induced on inputs of length k by each possible initialized recurrent net with architecture $\mathcal{A}$.

The questions Q1-Q4 posed above for static nets can also be asked, when suitably interpreted, for dynamic networks. Question Q1 leads to the topic of approximation of dynamical systems by recurrent nets. Work described in the PI papers [37] and [32] deals with this issue, mentioned in Section 3.4 of the report. Question Q2 has been researched in the recent PI papers [52], [24], and [27], while Q3 is touched upon in the first two of these references, and both are mentioned in Section 3.2.2.

2.5 Feedback Control Questions

Not all of our research in the neural network area deals with questions motivated by learning and generalization. A very different line of work of ours, which has achieved many important results yet needs further development, concerns the exploration of the capabilities of networks as controllers for nonlinear systems.

2.5.1 State-Feedback Control

Nonlinear control is often mentioned as one of the most promising areas of application for neural networks. We start with standard paradigm: a system of controlled differential equations

$$\dot{x} = f(x,u), \tag{19}$$

evolving in a manifold, where the right hand side is interpreted as a set of vector fields parameterized by the inputs (see e.g. [5] for precise definitions). Assume that x_0 is an equilibrium state ($f(x_0,u_0) = 0$ for some control value u_0); then one of the most basic control questions regards the existence of a feedback mapping $u = k(x)$, with $k(x_0) = u_0$, which renders the equilibrium x_0 of the closed loop system $\dot{x} = f(x,k(x))$ globally asymptotically stable. Most of the neurocontrol literature attempts to find such a feedback in the form of a neural network. (Of course, more complex control objectives are often posed, such as tracking with internal stability or model reference adaptive control, but we use stabilization, which is always a first design objective, to understand the problems in the "cleanest" possible setting.)

Very often, the stated objective is to obtain feedback laws implementable by nets of a special form, namely 1HL nets (recall Figure 3 and Equation (7)). This simple structure with clearly identifiable and tunable parameters is appealing from an adaptive control point of view. Moreover, the universality results for function approximation would seem to indicate that such networks are always sufficient. *However, this is very misleading.* It turns out that 1HL nets *are not* "universal" enough for the implementation of controllers, in a precise sense reviewed later. This was first pointed out in the paper [10], which also gave a general theorem showing that *two* hidden layers (and discontinuous activations) *are* sufficient. (The contribution was awarded a citation as an outstanding paper in the IEEE Transactions on Neural Networks). Intuitively, the reason for this apparent contradiction is that control problems are essentially inverse problems (one must solve for a trajectory satisfying certain boundary conditions). For such, as pointed out in subsection 2.1.1, more general types of networks are needed.

For an intuitive understanding of how this obstruction might occur, consider the following situation. Suppose that the objective is to globally asymptotically stabilize a planar system with respect to the origin using controllers $u{=}k(x)$ implementable by 1HL nets. It is easy to give examples for which there is no continuous feedback law that will achieve stabilization, even if the original system is very simple. Since its response is continuous, no 1HL net with continuous activations would be able to accomplish the stated objective. But what about allowing discontinuous σ such as Heaviside activations? (We ignore for now questions of possible nonexistence of solutions for ode's with discontinuous right-hand sides; the problem will be even more basic.) It turns out that even then it may be impossible to stabilize. Indeed, assume that we know some discontinuous feedback law $k_0(x)$ which stabilizes. It would appear that one can then obtain $k(x)$ simply by approximating k_0. However, as mentioned in subsection 2.1.1 and citing [10]), it is impossible in general to approximate a discontinuous k_0 *uniformly* by 1HL functions. But a weak type of approximation may not be enough for control purposes. For instance, it may be the case that for each approximant k_0 there is a small region Γ encircling the origin where the approximation is bad, in the sense the vector field $f(x, k_0(x))$ must point transversally *outward* everywhere on Γ, thus introducing an obstacle or "barrier" to global stabilization.

A precise result in given in the paper [10] for discrete-time systems (applicable to continuous-time via sample-and-hold control). That paper exhibits explicit examples of systems which are otherwise stabilizable but such that every possible feedback implementable by a 1HL net would fail, as any closed-loop system obtained in that way must give rise to a nontrivial periodic orbit. On the other hand, it also shows that if a system is stabilizable in any manner whatsoever, then it can also be stabilized using two-hidden layer nets.

To summarize, if stabilization requires discontinuities in feedback laws, it may be the case that no possible 1HL net stabilizes. Thus the issue of stabilization by nets is closely related to the standard problem of continuous and smooth stabilization of nonlinear systems, one that has attracted much research attention in recent years. Roughly, there is a hierarchy of state-feedback stabilization problems: those that admit continuous solutions, those that don't but can still be solved using 1HL nets with discontinuous activations, and more general ones (solvable with two hidden layers). It can be expected that an analogous situation will be true for other control problems. (Perhaps the reason that experimental neurocontrol papers have reported successes while using 1HL nets is that they almost always dealt with feedback linearizable systems. In the context of nonlinear systems, feedback linearizable ones constitute extremely restricted class, but they do admit continuous stabilizers, so the theoretical obstructions just discussed are not relevant.)

2.5.2 Saturated Linear Systems

The above limitations of 1HL nets in control notwithstanding, 1HL nets can be shown to be perfectly suited to *certain* control problems. One line of work by the PIs deals with the use of networks for the control of linear systems subject to actuator saturation. This is the situation modeled by equations of the following type:

$$\dot{x} = Ax + B\sigma(u)$$

where A and B are as usual in linear control theory and σ is a saturation such as tanh or π. It is often said that saturation is the most commonly encountered nonlinearity in control engineering, so the development of techniques for the control of such systems is obviously of great interest. Our work starts with the result by Fuller, around 1970, that it is in general impossible to globally stabilize the origin of such systems by means of linear feedback $u=Fx$ -for a more general result along those lines, see [4]- even if the system is open-loop globally controllable to the origin. This suggests the obvious question of searching for *non*linear feedback laws $u=k(x)$ that achieve such stabilization, and in particular for nicely behaved and easily implementable controllers (in contrast to optimal control techniques, which result in highly irregular feedback). In 1990 we proved that *smooth* stabilization is always possible. Motivated by our paper, Teel in 1992 showed that single-input multiple integrators can be stabilized by feedbacks which are themselves compositions of linear functions and iterated saturations. We were able soon after to extended Teel's result to arbitrary systems as above which are open-loop asymptotically controllable (equivalently, the rank of the matrix $[\lambda I - A, B]$ is n for all purely imaginary λ, and A has no eigenvalues with positive real part). More recently, we showed that one can *always use 1HL nets for implementing such feedback*; see [18]. The more recent work [29] extended these results to discrete time systems. As an example, the paper [43] developed an explicit design concerning a longitudinal flight model of an F-8 aircraft, with saturations on the elevator rate, and tested the obtained controller on the original nonlinear model. We chose the F-8 example since all parameters and typical trim conditions are publicly available, and the model has been often used as a test case for aircraft control designs. The procedure we followed consisted of first linearizing about an operating point and then constructing a globally stabilizing controller for the resulting linearization, with respect to this given trim condition, following the steps in our papers. Finally we proceed to compare the performance of the controller -applied to the original nonlinear airplane model and starting reasonably far from the desired operating point- with the "naive" controller that would result from applying a linear feedback law which would stabilize in the absence of saturations. The objective of the work was to show that the calculations in the abstract proofs can indeed be carried out explicitly (though this an extremely simple case compared to the generality of the results in our papers) and moreover, to study the performance of the resulting controller when used for the original, nonlinear, model. Although it turned out that our design provided unacceptable performance, the improvement with respect to linear feedback was remarkable, and we consider the results to be encouraging and indicating the usefulness of further work along this direction.

2.6 Recurrent Nets as Dynamical Systems

So far we have introduced in this overview several questions related to the areas of *learning* and of *control*. The former area provides a methodology for the study of different aspects of both feedforward and recurrent neural nets, and leads to the study of questions Q1-Q4 on approximation, learning dimensions, error surfaces, and computational complexity. Feedback control is an important application area, and leads to asking about the existence of networks implementing regulation laws with particular properties. We now look at third area, namely the study of properties of recurrent networks as *dynamical systems with inputs and outputs*. This leads to questions of observability, minimality, identification, and computational power. It turns out — perhaps surprisingly given the universal approximation properties possessed by recurrent nets — that such questions can be treated successfully to a large extent, in contrast to other classes of nonlinear systems more classically considered.

Observability, Identification

Recall (see e.g. [5]) that two states of a system are *distinguishable* if it is possible to tell them apart on the basis of input output experiments. The PIs wrote many papers since the late 1970s on such issues; references can be found in the ICM invited paper [35]. Observability means that every pair of distinct states must be distinguishable. Unobservability constitutes a fundamental limitation to the existence of an unbiased estimator of the internal states of a network. And from a synthesis rather than analysis point of view, the study of observability is one half of the study of redundancy: if there are indistinguishable states, one may in principle eliminate the redundancy and build a smaller network which achieves exactly the same performance objectives as specified in terms of input/output behavior. (The other half relates to the study of reachability.)

A formalization is as follows. An n-dimensional recurrent net $\Sigma = (A, B. C, \sigma)$ is *observable* if for each distinct $\xi \in \mathbb{R}^n$ and $\xi' \in \mathbb{R}^n$ it holds that

$$\lambda_{\Sigma,\xi} \neq \lambda_{\Sigma,\xi'}$$

(where Σ, ξ is the initialized recurrent net with initial state ξ). More precisely, one should define discrete-time or continuous-time observability, since the definition of the input/output mapping λ depends on the interpretation of the equations as difference or differential equations respectively. It turns out, interestingly, that there is a common algebraic characterization that applies in both cases. Assume that B satisfies this generic property: no row is identically zero and

$$\text{for each } i \neq j, \text{ there exists some } k \text{ such that } |b_{i,k}| \neq |b_{j,k}| \, , \tag{20}$$

and that (A, B, C, σ) is a homogeneous net whose activation is the standard sigmoid in Equation (1). Let $\mathcal{O}_{\rm c}(A, C)$ be the largest A-invariant coordinate subspace included in $\ker C$, where by a coordinate subspace we mean any subspace of $\mathbb{R}^n$ invariant under all the coordinate projections $\pi_i : \mathbb{R}^n \to \mathbb{R}^n$, $\pi_i e_j = \delta_{ij} e_i$ ($e_i, i = 1, \ldots, n$ denote the canonical basis elements in $\mathbb{R}^n$). It was shown in [15] that (A, B, C, σ) is observable if and only if $\ker A \bigcap \ker C = \mathcal{O}_{\rm c}(A, C) = 0$. This is a very simple characterization, easy to check algorithmically. One obtains as a corollary, under the above assumption on B, that the net is observable if the pair of matrices (A, C) is observable in the usual linear-algebraic sense (cf. [5]).

We have also studied in the past, and intend to continue, work on *parameter identifiability*, that is to say, the possibility of recovering the entries of the matrices A. B. and C, and, for initialized systems, also the initial state, from the input/output behavior of the net. Assume that we restrict attention to homogeneous nets that satisfy the following conditions. The activation σ is infinitely differentiable around zero, and satisfies the following mild nonlinearity assumption: $\sigma'(0) \neq 0$ and $\sigma^{(q)}(0) \neq 0$ for some $q > 2$. (So for analytic functions. we are just asking that σ be nonlinear and nonsingular at zero.) Furthermore, we assume that B satisfies (20) and no entry of B vanishes, and that the triple of matrices (A, B, C), $A \in R^{n \times n}$, $B \in R^{n \times m}$, $C \in R^{p \times n}$ is canonical. (That is, observable and controllable, as in [5], section 5.5; this is a generic set of triples, in the sense that the entries of those which do not satisfy the property form a proper Zariski-closed subset of $\mathbb{R}^{n^2+nm+np}$.) Then, in [14], a general result was proved, which in particular implies that if $\Sigma = (A, B, C, 0, \vec{\sigma}^{(n)})$ and $\Sigma' = (A', B', C'. 0, \vec{\sigma_1}^{(n)})$ have same input/output behaviors λ's, then $\sigma = \sigma'$ and the two triples of matrices are *sign-permutation equivalent* meaning that there exists a nonsingular matrix T such that $T^{-1}AT = A', T^{-1}B = B', CT = C'$, and T is a permutation matrix. That is, the two networks are exactly the same except for a relabeling of neurons.

This is a rather surprising result, which says that "function (input/output behavior) determines form (internal structure)". We recall that for linear systems the analogous result is far weaker, and only allows concluding equivalence modulo $GL(n, \mathbb{R})$ as opposed to modulo a discrete group (cf. [5]). The proof starts with the obvious idea of considering the parameters as extra constant states and computing the observables for the extended systems; this involves considering iterated Lie derivatives of observations under the vector fields defining the system and is a routine approach in nonlinear control. However, and this is the nontrivial part, extracting enough information from these Lie derivatives is not entirely easy, and the proof centers on that issue. Moreover, if the activation is analytic, the response to a *single* long-enough input function is theoretically enough for identification of all parameters, in analogy to the use of impulse-responses in linear systems theory. More recent work showed that, under additional assumptions on the activation (conditions which are satisfied when σ is the standard sigmoid, for example), one can obtain analogous results even for nonzero and distinct initial states (and one concludes that these states are also in the same orbit under the group action described above); see [41] for preliminary work (a full paper is still under preparation, as we wish to obtain as general a result as possible).

A related area is that of what we call *Fourier-recurrent* neural networks. These are networks with activations from the set $\{\sin x, \cos x\}$. (More precisely, we prefer to equivalently use complex state variables and the activation e^{ix}, but this is just a question of mathematical convenience.) In the paper [23], we presented a closed-form procedure, not involving any nonlinear optimization, for the identification of the entries of $A, B. C$ and the initial state for such nets. The procedure is based on Hankel-matrix techniques that are classical in the context of linear recurrences and their multivariable extensions developed in control theory (see a detailed discussion in Chapter 5 in [5]) as well as in many other areas such as, coding theory, for decoding BCH codes, and in learning theory, for sparse polynomial interpolation.

Computational Abilities; Computational Complexity

The last general subject that we wish to mention in this report deals with questions that are somewhat more abstract and are framed in the language of theoretical computer science. The topic is the exploration of the ***ultimate capabilities*** of recurrent nets viewed as *analog computing devices.* This area is a fascinating one, but very difficult to approach. Part of the problem is that, much interesting work notwithstanding, analog computation is hard to model, as difficult questions about precision of data and readout of results are immediately encountered –see the references in our papers cited below.

In a series of papers y one of the PIs and [17], as well as the April 28, 1995 issue of *Science*, and the "handbook" article [34]), we took the point of view that artificial neural nets provide an opportunity to reexamine some of the foundations of analog computation from the new perspective afforded by an extremely simple yet surprisingly rich model, in a context where techniques from dynamical systems theory interact naturally with more standard notions from theoretical computer science. For recurrent nets with activation the piecewise linear function π in Equation 2, we derived results on deterministic versus nondeterministic computation, and related the study to standard concepts in complexity theory. Perhaps the previous work closest in spirit is that on real-number-based computation started by Blum, Shub, and Smale ("BSS" model). In contrast to that line of work, however, recurrent nets do not incorporate discontinuous (and hence highly nonrobust), infinite precision "if-then-else" decisions, nor can discrete results be read out of the system through infinite precision measurements. Thus recurrent nets are more restricted than the BSS model.

One of the most unexpected conclusions was that, at least within the formalism of analog computation proposed there, recurrent neural nets (with activation π) are a *universal* model, in much the same manner as Turing machines are for classical digital computing. It was shown that no possible "analog computer" –in a sense precisely defined– could ever have more power –again in a precisely defined sense– up to polynomial time speedups. Particularly satisfying theoretically is the fact that the most natural categories for the values of weights correlate perfectly with different natural subclasses of computing devices, and can be summarized in an extremely elegant fashion: integer weights correspond to finite automata, rational to Turing machines, and real to arbitrary "P/poly" computation (the well-studied Karp-Lipton class consisting of nonuniform polynomial-size circuits or equivalently sparse-oracle Turing Machines). (Furthermore, we showed in [38] how to characterize computations by networks having weights in certain Kolmogorov-complexity definable classes in between rationals and reals.)

Another unanticipated –and intriguing– conclusion was that the class NP of nondeterministic polynomial-time digital computation is *not* included in what can be computed in polynomial time with analog devices (this is proved under standard assumptions of the "P≠NP" type). Thus the solution of combinatorial problems using analog devices may be subject to the same ultimate computational obstructions, for large problem sizes, as with digital computing.

The work on computation capabilities turns out to be relevant to the study of "hybrid" control systems which combine automata and linear systems, or saturation devices in feedback loops. For example, the problem of determining if a dynamical system $x(t+1) = \vec{\sigma}^{(n)}(Ax(t))$ ever reaches an equilibrium point, from a given initial state, is shown to be effectively undecidable (at least for $\sigma{=}\pi$) as a consequence of the established universality. (In Hopfield-type nets, when dealing with content-addressable retrieval, the initial state is taken as the "input pattern" and the final state, if there is convergence in finite time, as a class representative.) The implications of this result to various questions in control theory and in particular hybrid systems are discussed in some detail in the papers [51] and [36].

3 Some More Details

In this part of the report, we have chosen to provide some more details on a few selected subtopics, especially those represented in very recent work and hence possibly less easily available. Once more we remind the reader that the Web page found at the URL: **http://www.math.rutgers.edu/~sontag/** provides access to a substantial number of papers by the PIs, to be consulted for details and a mathematically rigorous presentation.

3.1 Error Functions

Here we deal with question **Q4**: "what are the properties of the error function being minimized in (17)?", for hypotheses classes of the form $\mathcal{F} = \mathcal{F}_{\mathcal{A}}$. This is motivated in our development by the risk minimization solution of the learning problem, but it is of course what is done experimentally, not necessarily with any theoretical justification, in standard neural net practice: given a parametric form, find parameters so that the error on the training data is minimized. Numerical algorithms based on gradient descent (so-called "backpropagation") are usually employed for the minimization. Thus it is necessary to study the points at which the derivative of this error function may become zero, and more particularly the location of possible local but nonglobal minima. Among PI work in this direction are the references [7], [9], [50], [22]. We now summarize some facts from the last of these papers. The emphasis of this past work was on 1HL nets, which, as discussed in Section 2.1.1, are those most commonly found in applications.

By a 1HL architecture we mean an homogeneous, scalar-output, single hidden layer architecture with fully connected layers, that is, $\mathcal{A} = (\mathcal{L}^1, \mathcal{L}^2)$ and I^i and J^i, $i = 1, 2$ are both matrices with all entries equal to one. If $\mathcal{A}$ has n neurons and activation σ, then the nets with architecture $\mathcal{A}$ are precisely the possible 1HL nets with n neurons and activation σ. These compute functions as in Equation (7), can be represented diagrammatically as in Figure 3, and the parameter space has dimension $r = n(m+2)+1$, where $\rho \in \mathbb{R}^r$ represents the scalars $c_0, \ldots, c_n$ and $a_1, \ldots, a_n$, and the m-row vectors $A_1, \ldots, A_n$. For purposes of this discussion, we will assume that $\sigma(x) = \tanh(x)$ (or up to rescalings, the standard sigmoid), which is the activation most commonly used in experimental work, and we assume that n has been fixed. (The study of good choices for n leads to "structural risk minimization" studies, about which we do not have anything to contribute at this time.)

There are portions of the parameter space that give rise to degeneracies. For instance, if one coefficient c_i ($i \neq 0$) vanishes, then the input/output function f is independent of the values of the corresponding A_i and a_i. If some $A_i = 0$ then the corresponding term is constant and can be absorbed into c_0. If for two different $i \neq j$ it is the case that $A_i = A_j$ and $a_i = a_j$, then the terms corresponding to i and j can be combined, and only the sum $c_i + c_j$ is relevant, resulting also in a loss of dimensionality. Similarly, since tanh is an odd function, if $(A_i, a_i) = -(A_j, a_j)$ then terms can be combined as well. Thus a more natural parameter space is the subset consisting of all the a_i's, c_i's, and A_{ij}'s for which these exceptional situations do not occur. We will restrict attention to parameters in this subset.

Assume given a training or regression data set ("labeled sample") $\nu = (u_1, y_1), \ldots, (u_s, y_s)$, where we interpret the u_i's as input vectors ("regressors" in statistical terms) and the scalars y_i's as targets or response vectors desired for the respective u_i's. The (regression) problem is that of minimizing (typically by means of steepest descent or other local search techniques) the quadratic loss, error, or "risk" $\mathrm{Emp}\,(f, \nu)$ over all $f = f(\rho, \cdot) \in \mathcal{F}_{\mathcal{A}}$, that is to say, over all $\rho \in \mathbb{R}^r$. We write $E^{\nu}(\rho)$ instead of $\mathrm{Emp}\,(f(\rho, \cdot), \nu)$ in order to emphasize the dependence of this quantity on the parameters, once that the labeled sample has been fixed. Thus, the problem is to study the function

$$E^{\nu}(\rho) := \frac{1}{2}\sum_{i=1}^{s}\Big(f(\rho,u_i)-y_i\Big)^2$$

with domain the above parameter set.

Critical Point Analysis

It has been often remarked that, even for extremely simple cases (such as n=1 and supposing that the inputs are binary vectors) there arise critical points associated to non-global local minima, and thus the study of the set of critical points of E^{ν} has been frequently put forward as a research topic. In this context it has also been observed several times that —as with other least-squares problems— pathological behavior will depend heavily on the training sets not being in "general position" in appropriate senses of probability or topological density. In the paper [22] we obtained several characterizations of the critical set.

One of the main results given in that paper was that the set of critical points is finite, and in particular less than $2^{8(sn)^2}$ (assuming that there are enough samples to make the problem not underdetermined, specifically that $s \geq 2n(m+2)+3$, and for generic regression data). If the number of samples scales linearly on the number of nodes n, and assuming a constant input dimension, an upper bound of the type

$$2^{cn^4}$$

results. A lower bound of the type $2^{c'n\log n}$ also holds, due to symmetries in the problem: any exchange among the n terms in the sum preserves f.

We review next the organization of the proof, to illustrate the types of techniques used. We first showed that analytically parameterized classes of functions can be identified generically on the basis of just $2r+1$ samples, if r is the number of free parameters. This part of the paper depended on basic facts about real-analytic functions, all consequences of the basic stratification theorems for subanalytic sets due to Hironaka, Lojasiewicz, and others (see [1] for an introduction, with complete proofs, to the needed results on stratification theory). We then studied critical points for least-squares error criteria; this step relied upon elementary differential topology, basically the elementary Morse theory found in most textbooks, but applied only after yet another application of analytic set theory. We then showed, for generic analytic problems, the countability of the set of functions giving rise to critical parameter values. and a refinement showing that this set is in fact finite, provided that the parametric class of functions be definable logically in terms of the exponential and certain other special analytic functions; this was based on the recent work in logic, dealing with o-minimal logical theories, due to Wilkie, MacIntyre. van den Dries, Miller, Gabrielov, and others. Finally, we specialized to single hidden layer networks, where one can use results on determination of parameters (as given in [31]) in order to obtain finiteness of the set of critical parameters. Once that finiteness is known, one may use Khovanskii estimates on Betti numbers for sets defined by exponential/algebraic expressions in order to bound the number of connected components, since the set in question is now known to be an embedded submanifold, and this bound is of course just the number of points, giving the estimate mentioned above.

Convergence to True Parameters

In the above context of minimization of the a loss criterion, one needs to understand the convergence of gradient descent procedures (and of related "on line" methods, which only approximate the gradient by considering one term at a time). In this context, we mention that in general the existence of nonglobal local minima means of course that no global convergence may be expected. For a special case treated in [9], it is possible to show that gradient descent converges globally (we omit the details here) but in the general case the existence of such obstructions is unavoidable. Local convergence will hold if the number of inputs is large enough (the bound given above is sufficient), and in that case one may ask the following question: if the training data was indeed generated by a 1HL net (seen as a "black box"), does gradient descent lead to the true parameters? This issue, for the generic parameter space mentioned above, was posed (in slightly different language, but the same mathematical problem) in the late 1980s by Hecht-Nielsen. A positive answer follows from the results presented in [2], for the special case of the activation tanh. The more recent work [31], mentioned above, extended this to a wide class of real-analytic functions, using complex-variable techniques rather than the asymptotic analysis of [2].

3.2 Estimation of Learning Dimension

Here we deal with question **Q2**: "does $\mathcal{F}$ have finite pseudo-dimension?", for hypotheses classes of the form $\mathcal{F} = \mathcal{F}_{\mathcal{A}}$. Recall the PAC learning formulation discussed in Section 2.4.1: a set of labeled training samples is provided, and a network must be obtained which is then expected to ("probably and approximately") correctly classify previously unseen inputs. In this context, a central problem is to estimate the amount of training data needed to guarantee satisfactory learning performance, and this in turn leads to the search for pseudodimension estimates.

We first define the special case of VC dimension, and then pseudo- (or Pollard) dimension. The concept of VC dimension is classically defined in terms of abstract concept classes; we review that first and then interpret in terms of functions. As in Section 2.4, we are given an *input set* $\mathcal{U}$. Assume also given a family of subsets $\mathcal{C}$ of $\mathcal{U}$, called the set of "concepts." A subset $\mathcal{U}_0 \subseteq \mathcal{U}$ is said to be *shattered* (by the class $\mathcal{C}$) if for each subset $B \subseteq \mathcal{U}_0$ there is some $C \in \mathcal{C}$ such that $B = C \bigcap \mathcal{U}_0$. The VC dimension is then the largest possible positive integer κ (possibly $+\infty$) so that there is some $\mathcal{U}_0 \subseteq \mathcal{U}$ of cardinality κ which can be shattered. An equivalent manner of stating these notions, somewhat more suitable for our purposes, proceeds by identifying the subsets of $\mathcal{U}_0$ with Boolean functions from $\mathcal{U}_0$ to $\{0,1\}$: to each such Boolean function ϕ there is an associated subset, namely $\{x \in \mathcal{U}_0 \,|\, \phi(x) = 1\}$, and conversely, to each set $B \subseteq X$ one can associate its characteristic function ϕ_B defined on the set $\mathcal{U}_0$. Similarly, we can think of the sets $C \in \mathcal{C}$ as Boolean functions on $\mathcal{U}$ and the intersections $C \bigcap \mathcal{U}_0$ as the restrictions of such functions to $\mathcal{U}_0$. Thus we restate the definitions now in terms of functions.

Given the set $\mathcal{U}$, and a subset $\mathcal{U}_0$ of $\mathcal{U}$, a *dichotomy* on $\mathcal{U}_0$ is a function $\delta : \mathcal{U}_0 \to \{0,1\}$. Assume given a class $\mathcal{F}$ of functions $\mathcal{U} \to \{0,1\}$, to be called the class of *classifier* functions. The subset $\mathcal{U}_0 \subseteq \mathcal{U}$ is *shattered* by $\mathcal{F}$ if each dichotomy on $\mathcal{U}_0$ is the restriction to $\mathcal{U}_0$ of some $\phi \in \mathcal{F}$. The *Vapnik-Chervonenkis (VC) dimension* $\mathrm{VC}\,(\mathcal{F})$ is the supremum (possibly infinite) of the set of integers κ for which there is some subset $\mathcal{U}_0 \subseteq \mathcal{U}$ of cardinality κ which can be shattered by $\mathcal{F}$.

By abuse of terminology, when $\mathcal{F}$ is a class of real-valued functions (as with maps $\mathcal{F}_{\mathcal{A}}$ induced by fixed neural architectures), *by* $\mathrm{VC}\,(\mathcal{F})$ *one means* $\mathrm{VC}\,(\mathcal{H}(\mathcal{F}))$, where $\mathcal{H}(\mathcal{F}) = \{\mathcal{H} \circ f, f \in \mathcal{F}\}$, and H is the Heaviside function introduced earlier. Thus $\mathrm{VC}\,(\mathcal{F})$ quantifies the amount of training data needed for reliable prediction when using functions in $\mathcal{F}$ as classifiers: a positive output means "accepted" and a negative output to "rejected". This is consistent with the standard use of neural networks as classification machines. For binary-valued hypotheses classes, the VC dimension provides tight estimates (as opposed to merely upper bounds) of the sample complexity discussed in Section 2.4.1, assuming we are interested in learning with respect to the class of all possible (Borel) probability measures: under certain simple nontriviality assumptions and for $0 < \varepsilon < \frac{1}{2}$,

$$s(\varepsilon, \delta) \geq \max\left\{ \frac{1-\varepsilon}{\varepsilon} \ln\left(\frac{1}{\delta}\right), \text{VC}(\mathcal{F})(1 - 2(\varepsilon(1-\delta)+\delta)) \right\} .$$

Together with the upper bounds reviewed earlier, this says roughly that sample complexity is proportional to $\text{VC}(\mathcal{F})$.

One may also define a measure useful for learning real-valued functions, as discussed in Section 2.4.1. The notion of pseudo-dimension is used in the framework developed by Haussler and based on previous work by Vapnik, Chervonenkis, and Pollard. Given a class of functions $\mathcal{F}$ from $\mathcal{U}$ to $\mathcal{Y}$, we may introduce, for each $f \in \mathcal{F}$, the function

$$q_f \; : \; \mathcal{U} \times \mathcal{Y} \to \{0,1\} \; : \; (x,y) \mapsto \text{sign}\,(f(x) - y)$$

as well as the class $\mathcal{F}_0$ consisting of all such q_f. The ***pseudo-dimension*** of $\mathcal{F}$ is the extension of VC dimension to non-binary function classes given by the formula $\text{PD}(\mathcal{F}) := \text{VC}(\mathcal{F}_0)$. We will focus here on VC dimension questions, that is, binary classification, but many of our results (the upper bounds by analogous proofs, and the lower bounds as corollaries) apply also to pseudo-dimension estimates. (One should observe that, in contrast to the binary case, finiteness of pseudo-dimension is merely a sufficient, not a necessary, condition, for learnability. On the other hand, the actual proofs provide also lower bounds for so-called scale-sensitive dimensions, which for the continuous output case do provide necessary conditions. Another important observation is that boundedness of the output space is a requirement in the connection between pseudo-dimension and PAC learnability, which means that results must either assume a "squashing" of the output, or consider bounded loss functions such as $|y_1 - y_2|^2 / (1 + |y_1 - y_2|^2)$.)

3.2.1 VC Dimension for Feedforward Nets

The well-known work of Cover in 1968 and Baum and Haussler in 1989 dealt with the computation of $\text{VC}(\mathcal{F})$ when the class $\mathcal{F}$ consists of networks built up from hard-threshold activations and having r weights; they showed that $\text{VC}(\mathcal{F}) = O(r \log r)$. Conversely, Maass showed in 1993 that there is also a lower bound of this form. It would appear that this definitely settled the VC dimension (and hence also the sample size) question. However, that estimate assumed an architecture based on hard-threshold (Heaviside) activations. In contrast, the usually employed gradient descent learning algorithms ("backpropagation" method) rely upon *continuous* activations, that is, neurons with graded responses. As pointed out in [8], the use of analog activations, which allow the passing of rich (not just binary) information among levels, may result in higher memory capacity as compared with threshold nets. This has serious potential implications in learning, essentially because more memory capacity means that a given function f may be able to "memorize" in a "rote" fashion too much data, and less generalization is therefore possible. Indeed, the paper [11] showed that there are conceivable (though not very practical) neural architectures with extremely high VC dimensions. Thus the problem of studying $\text{VC}(\mathcal{F})$ for analog networks is an interesting and relevant issue. Two important contributions in this direction were the papers by Maass in 1993 and by Goldberg and Jerrum in 1993. which showed upper bounds on the VC dimension of networks that use piecewise polynomial activations. The paper [39] introduced techniques from model theory and analytic function theory to show that the VC dimension is finite for large classes of activations (including the standard sigmoid, in particular), and following along this direction a paper by MacIntyre and Karpinski recently succeeded in establishing an explicit upper bound for the standard sigmoid as well as other Pfaffian activations.

Goldberg and Jerrum established for piecewise polynomial activations an upper bound of $O(r^2)$, where, as before, r is the number of weights. However it was an open problem in a 1994 survey by Maass if there is a matching r^2 lower bound for such networks, and more generally for arbitrary continuous-activation nets. It could have been the case that the upper bound $O(r^2)$ is merely an artifact of the method of proof, and that reliable learning with continuous-activation networks is still possible with far smaller sample sizes, proportional to $O(r \log r)$.

But this is not the case, and in the papers [25], [53] we answered Maass' open question in the affirmative. As in [11], we say that the activation $\sigma : \mathbb{R} \to \mathbb{R}$ is *sigmoidal*, or *a sigmoid*, if: (1) σ is differentiable at some point x_0 where $\sigma'(x_0) \neq 0$ and (2) $\lim_{x \to -\infty} \sigma(x) = 0$ and $\lim_{x \to +\infty} \sigma(x) = 1$ (the limits 0 and 1 can be replaced by any distinct numbers). (Note that the first condition rules out the Heaviside activation). Then there are architectures with arbitrary large numbers of weights r and VC dimension proportional to r^2. The proof relies on first showing that networks consisting of two types of activations, Heavisides and linear, already have this power. This is a somewhat surprising result, since purely linear networks result in VC dimension proportional to r, and purely threshold nets have, as per the results quoted above, VC dimension bounded by $r \log r$. The desired result on continuous activations is then obtained, approximating Heaviside gates by σ-nets with large weights and approximating linear gates by σ-nets with small weights (sigmoids are "locally linear and globally thresholds"). A number of variations, dealing with Boolean inputs, or weakening the assumptions on σ, are also discussed in [25], whose last section also describes an interpretation of the results in terms of threshold-only networks with "shared" weights. Our result applies, as a very special case, to the standard sigmoid $1/(1 + e^{-x})$.

3.2.2 VC Dimension for Recurrent Nets

We explained in Section 2.4.3 the interest in hypotheses classes whose inputs $u \in \mathcal{U}$ are themselves finite sequences, and specifically we wish to look at classes $\mathcal{F}_{\mathcal{A},k}$ (see Equation (15)) consisting of those mappings $(\mathbb{R}^m)^k \to \mathbb{R}$ induced on inputs of length k by the possible initialized recurrent net with a given architecture $\mathcal{A}$. We now describe some results from the papers [24] and [27]. In all our results, we will take the number of input components $m = 1$, for simplicity, and we consider only homogeneous (all activations equal) architectures. We assume we are working in discrete time and interpret Equations (9) as difference equations. By σ-architecture, we mean an architecture where all activations are the same function $\sigma : \mathbb{R} \to \mathbb{R}$. (The choice of $m = 1$ makes our lower bounds more interesting. It is fairly easy, though notationally somewhat more cumbersome, to extend the upper bounds to vector inputs. The same can be said about the homogeneity assumption.)

Given any n-dimensional architecture $\mathcal{A}$ with $m = p = 1$, and any $k > 0$, we denote $\text{VC}(\mathcal{A}, k) = \text{VC}(\mathcal{F}_{\mathcal{A},k})$ and refer to this quantity also as the "VC dimension of $\mathcal{A}$ when receiving inputs of length k". We write r for the total number of parameters in the architecture.

We are particularly interested in understanding the behavior of $\text{VC}(\mathcal{A}, k)$ as $k \to \infty$, for various recurrent architectures, as well as the dependence of this quantity on the number of weights and the particular type of activation being used.

The first case of interest is that in which σ is the identity. This means that one is using linear dynamical systems as learners in the sense of PAC theory (or, for VC dimension, the sign of the output of such a system, which represents the simplest possible quantization of the output signal). Comparing with classical "perceptrons", the maps in $\mathcal{F}_{\mathcal{A},k}$ represent in this case inner products with a separating vector in $\mathbb{R}^k$ that is the impulse-response of a recursive digital filter of order n. Seen in this context, the usual perceptrons are nothing more than the very special subclass of "finite impulse response" systems (all poles at zero); thus it is appropriate to call the more general class "recurrent" or "IIR (infinite impulse response)" perceptrons. One may reliably predict, for given accuracy and confidence parameters, on the basis of $O(k)$ samples, but this is too conservative if there is reason to believe that the data may be linearly separable by a low-dimension dynamical system, that is, if we are interested in learning using the above hypothesis class and $k \gg n$. Roughly speaking, the main result in [24] was that the number of samples needed is proportional to the *logarithm* of the length k (as opposed to k itself, as would be the case if one did not take advantage of the recurrent structure). The upper bounds are obtained using simple arguments from algebraic geometry, while the lower bounds involve an apparently result on dual VC dimensions, for which we needed to develop the theory. The main result. for the non-sparse case in which we assume that all entries of the matrices are parameters (actually, the parameter space is redundant, and one may assume, using appropriate canonical forms. that $r = 2n$), is, more precisely, as follows. We write $\textsc{vc}\,(n,k)$ instead of $\textsc{vc}\,(\mathcal{A},k)$, where $\mathcal{A}$ is the architecture being discussed. Then:

$$\max\left\{n, n\lfloor\log(\lfloor\frac{k-1}{n}\rfloor)\rfloor\right\} < \textsc{vc}\,(n,k) \leq \min\{k\,,\,20n + 4n\log(k-n+1)\}\ .$$

Observe that this means, in particular, that when $k > \max\{n^2, 32\}$ it holds that $\frac{n}{2}\log k \leq \textsc{vc}\,(n,k) \leq 8n\log k$.

By a ***threshold*** recurrent architecture we mean a homogeneous one with $\sigma = \mathcal{H}$. Our main results in [27], ignoring multiplicative constants, say roughly the following: (1) For architectures with activation σ = any fixed nonlinear polynomial, the VC dimension is $\approx rk$, and the lower bound holds for any sigmoidal activation (but there are no possible upper bounds that hold for arbitrary sigmoids). (2) For architectures with activation σ = any fixed *piecewise* polynomial, the VC dimension is between rk and r^2k. (3) For architectures with activation $\sigma = \mathcal{H}$ (threshold nets), the VC dimension is between $r\log(k/r)$ and $\min\{rk\log rk, r^2 + r\log rk\}$. (4) For the standard sigmoid $\sigma(x) = 1/(1+e^{-x})$, the VC dimension is between rk and r^4k^2. Upper bounds are obtained by a combination of "unfolding" (and application of bounds known for feedforward nets) and ad-hoc results. The lower bounds are obtained by means of explicit examples of architectures which perform various bit-decoding operations on weights. In the sigmoidal case one uses again the "locally linear and globally threshold" property. It is possible to generalize the lower bound that holds in the sigmoidal case to even more arbitrary activations. Let σ be a function which is twice continuously differentiable function in an open interval containing some point x_0 where $\sigma''(x_0)\neq 0$. The VC dimension of recurrent architectures with activation σ, with r weights and receiving inputs of length k, is also $\Omega(rk)$. The construction in this case is based on ideas from symbolic dynamics, essentially using a chaotic-type system to decode weight information and hence affect the progress of the computation.

It is interesting to contrast the situation with the one that holds for feedforward nets. For the latter, it holds, in general terms, that linear activations provide VC dimension proportional to r, threshold activations give VC dimension proportional to $r\log(r)$, and piecewise polynomial activations result in VC dimension proportional to r^2.

3.2.3 Other Dimensions

As we discussed, for feedforward nets the VC dimension grows in general at least as fast as the *square* r^2 of the number of adjustable weights r. This fact can be seen as optimistic (relatively low sample complexity), but the bound is more pessimistic than some experimental data would seem to indicate. It is essentially impossible to design experiments for testing PAC learning, because of the need to estimate prediction confidence for large families of distributions; most experiments deal with specific distributions. Because of this, the first PI was asked, at the end of his plenary-talk presentation of the results from [25] and [53] at NIPS'95, if it was not possible that sets of input patterns which can be shattered are all in some sense "special" and that if we ask instead, as done in the classical literature in pattern recognition, for the shattering of *all sets in "general position"* (as in Cover's work on capacity of perceptrons), then an upper bound of $O(r)$ might hold. After further research, we produced the paper [26], which answered this in the affirmative. We established a linear upper bound for arbitrary sigmoidal (as well as threshold) neural nets for what we may call a "generic shattering dimension". We omit the details here, but wish to emphasize that one major direction for future work is that of investigating the relevance of our generic shattering dimension to variations of PAC learning, when one weakens the requirement that generalization capabilities must hold with respect to all possible input distributions. (The upper bound is also useful in the very different context of understanding computational abilities; as an illustration of this fact, we mention that our main result, announced electronically, was immediately employed by Maass in a new paper contrasting the computational power of spiking neurons with that of sigmoidal neural networks.)

3.3 Approximations

Here we deal with question **Q1**: "How large can the errors $\underline{\mathrm{Err}}(P)$ be?", for hypotheses classes of the form $\mathcal{F} = \mathcal{F}_{\mathcal{A}}$. Recall that this question characterizes the minimum potentially achievable error, no matter how many samples are seen or how powerful an algorithm is used. The smaller $\underline{\mathrm{Err}}(P)$, the more useful is the estimate in Equation (16). Take the case when the underlying probability densities in the general learning paradigm are induced from input output data of the form $(u, g(u))$, where g is an unknown target function whose behavior we are attempting to emulate by means of hypotheses $f \in \mathcal{F}$. In that case, we are led to questions of function approximation. With reasonable prior assumptions on the possible g's, one tries to obtain good estimates of the best possible approximation error $\inf_{f\in\mathcal{F}} \|g - f\|$, for various function space norms on $\mathcal{F}$. That is, one wants to study the distance to $\mathcal{F}$ of elements g which satisfy the prior assumptions.

3.3.1 Feedforward Nets

In the neural nets literature, one finds a claim to the extent that approximations by means of (1HL) neural networks may require less parameters than conventional techniques. What is meant by this is that approximations of functions in certain classes (defined typically in harmonic analysis terms or, say, the unit ball of a Sobolev space W_p^r) to within a desired error tolerance can be obtained using "small" networks. In contrast, the argument goes, using for instance orthogonal polynomials, splines, or Fourier series, would require an astronomical number of terms, especially for multivariate inputs. Unfortunately, this claim represents a misunderstanding of the very nice results obtained by Andrew Barron and Lee Jones during the last few years. Their results apply in principle also to give efficient approximations with various types of classical basis functions as long as the basis elements can be chosen in a nonlinear fashion (just as is the case with neural networks). For instance, splines with free (rather than fixed) nodes, or trigonometric series with adaptively selected frequencies, will have the same properties. What is important is the possibility of *selecting* terms adaptively, in contrast to the use of a large basis containing many terms and fitting these through the use of least squares. Thus, the important fact about these results is that they emphasize that *non*linear parameterizations may require less parameters than linear ones to achieve a guaranteed degree of approximation. (Abstractly, this is not so surprising: for an analogy, consider the fact that a even a one-parameter analytic curve, based on ergodic motions, can be used to approximate arbitrarily well every element in an Euclidean space $\mathbb{R}^d$, but no $(d-1)$-dimensional subspace can do so.) For an exposition and a precise theorem comparing rates of approximation by such neural net and nonlinear adaptation approaches vs. rates obtainable with classical approximation techniques, the reader may wish to consult the paper [3]. (Constraints on nonlinear adaptation procedures also exist, and are based on the theory of nonlinear N-widths, but we do not discuss that subject here.) The paper [21] dealt with some of these questions regarding rates of approximation, and we review some of the setup now.

The subject of that paper concerns the problem of approximating elements of a Banach space X – typically presented as a space of functions – by means of finite linear combinations of elements from a predetermined subset S of X. In contrast to classical linear approximation techniques, where optimal approximation is desired and no penalty is imposed on the *number* of elements used, we are interested there in *sparse* approximants, that is to say, combinations that employ few elements. In particular, we are interested in understanding the rate at which the achievable error can be reduced as one increases the number allowed. Such questions are of obvious interest in areas such as signal representation, numerical analysis, and neural networks. In that latter context, if we wish to study approximations by input/output maps of 1HL networks with n hidden units, one must consider linear combinations of n-element subsets of

$$S = \{g : \mathbb{R}^m \to \mathbb{R} \mid \exists A \in \mathbb{R}^m, a \in \mathbb{R}, \mid g(u) = \pm\sigma(A \cdot u + a)\},$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation of interest. 1HL approximations are of interest because of the results which insure that, for each compact subset M of $\mathbb{R}^m$, restricting elements of S to M, the closed linear span of S is all of $C^0(M)$ (under extremely weak conditions on σ; being locally Riemann integrable and non-polynomial is enough).

Rather than arbitrary linear combinations $\sum_i c_i g_i$, with c_i's real and g_i's in S, it turns out to be easier to understand approximations in terms of combinations that are subject to a prescribed upper bound on the total coefficient sum $\sum_i |c_i|$. After normalizing S and replacing it by $S \cup -S$, one is led to studying approximations in terms of convex combinations. This is the focus of [21]. To explain the previous results and new contributions, we first introduce some notation. Let X be a Banach space, with norm $\|\cdot\|$. Take any subset $S \subseteq X$. For each positive integer n, we let $\mathrm{co}_n S$ consist of all sums $\sum_{i=1}^n c_i g_i$, with $g_1, \ldots, g_n$ in S and reals $c_i \in [0,1]$, $\sum_i c_i = 1$. The distance from an element $f \in X$ to this space is denoted $\|\mathrm{co}_n S - f\| := \inf\{\|h - f\|, h \in \mathrm{co}_n S\}$. Let ϕ be a positive function on the integers. We say that the space X admits a *(convex) approximation rate* $\phi(n)$ if for each bounded subset S of X and each $f \in \overline{\mathrm{co} S}$, $\|\mathrm{co}_n S - f\| = O(\phi(n))$. Jones and Barron showed that every Hilbert space admits an approximation rate $\phi(n) = 1/\sqrt{n}$. One of our main objectives in [21] was the study of such rates for non-Hilbert spaces. (Barron in 1992 did show that the same rate is obtained in the uniform norm, but only for approximation with respect to special classes of sets S.) Spaces L^p with p equal to or slightly greater than one are particularly important because of their usefulness for robust estimation, and there have been experimental results for regression with neural networks, showing the superiority of L^p $(p \ll 2)$ to L^2 in that context (see [21]).

Another issue of interest is as follows. Jones considered the procedure of constructing approximants to f incrementally, by forming a convex combination of the last approximant with a *single* new element of S; in this case, the convergence rate in L^2 is interestingly again $O(1/\sqrt{n})$.

Incremental approximants are especially attractive from a computational point of view. In the neural network context, they correspond to adding one "neuron" at a time to decrease the residual error. We next define this concept precisely. Again let X be a Banach space with norm $\|\cdot\|$. Let $S \subseteq X$. An *incremental sequence* (for approximation in $\operatorname{co} S$) is any sequence $f_1, f_2, \ldots$ of elements of X so that $f_1 \in S$ and for each $n \geq 1$ there is some $g_n \in S$ so that $f_{n+1} \in \operatorname{co}(\{f_n, g_n\})$. We say that an incremental sequence $f_1, f_2, \ldots$ is *greedy* (with respect to $f \in \overline{\operatorname{co} S}$) if $\|f_{n+1} - f\| = \inf\{\|h - f\| \mid h \in \operatorname{co}(\{f_n, g\}), g \in S\}$, $n = 1, 2, \ldots$. The set S is generally not compact, so we cannot expect the infimum to be attained. Given a positive sequence $\epsilon = (\epsilon_1, \epsilon_2, \ldots)$ of allowed "slack" terms, we say that an incremental sequence $f_1, f_2, \ldots$ is *ε-greedy* (with respect to f) if $\|f_{n+1} - f\| < \inf\{\|h - f\| \mid h \in \operatorname{co}(\{f_n, g\}), g \in S\} + \varepsilon_n$, $n = 1, 2, \ldots$. Let ϕ be a positive function on the integers. We say that S ***has an incremental (convex) scheme with rate*** $\phi(n)$ if there is an incremental schedule ε such that, for each f in $\overline{\operatorname{co} S}$ and each ε-greedy incremental sequence $f_1, f_2, \ldots$, it holds that $\|f_n - f\| = O(\phi(n))$ as $n \to +\infty$. Finally, we say that the space X ***admits incremental (convex) schemes with rate*** $\phi(n)$ if every bounded subset S of X has an incremental scheme with rate $\phi(n)$. The intuitive idea behind this definition is that at each stage we attempt to obtain the best approximant in the restricted subclass consisting of convex combinations $(1 - \lambda_n)f_n + \lambda_n g$, with λ_n in $[0, 1]$, g in S, and f_n being the previous approximant. It is also possible to select the sequence $\lambda_1, \lambda_2, \ldots$ beforehand. We say that an incremental sequence $f_1, f_2, \ldots$ is ϵ-greedy (with respect to f) with ***convexity schedule*** $\lambda_1, \lambda_2, \ldots$ if $\|f_{n+1} - f\| < \inf\{\|((1 - \lambda_n)f_n + \lambda_n g) - f\| \mid g \in S\} + \varepsilon_n$, $n = 1, 2, \ldots$.

The main objective of the paper [21] was to analyze both optimal and incremental rates in broad classes of Banach spaces, specifically including L^p, $1 \leq p \leq \infty$. It is a triviality that optimal approximants to approximable functions always converge. However. the rates of convergence depend critically upon the structure of the space. In some spaces, like L^1, there exist target functions for which the rate can be made arbitrarily slow. In Banach spaces of (Rademacher) type t with $t > 1$, however, a rate bound of $O(n^{-1+1/t})$ is obtained. For L^p spaces these results specialize to these ("no" means that the approximants do not always converge):

p	1	(1,2)	$[2, \infty)$	∞
optimal	1	$n^{-1+1/p}$	$n^{-1/2}$	1
incremental	NO	$n^{-1+1/p}$	$n^{-1/2}$	NO

Particular examples of L^p spaces are given to show that the orders given in our bounds cannot in general be sharpened. In the incremental case, a particularly interesting aspect of the results is that the new element of S added to the incremental approximant is not required to be the best possible choice. Instead, the new element can meet a less stringent test, and the convex combination of the elements included in the approximant is not optimized. Instead a simple average is used. (This is an example of a fixed convexity schedule. Thus, our incremental approximants are the simplest yet studied, simpler even than those of Jones. Nonetheless, the same worst-case order is obtained for these approximants on L^p, $1 < p < \infty$, as for the optimal approximant. In more general spaces, the incremental approximants may not even converge. However, if the space has a modulus of smoothness of power type greater than one, or is of Rademacher type t, then rate bounds can be given. The results are based on explicit constructions as well as use of the rich theory developed by Lindenstrauss on smoothness moduli and by Ledoux, Talagrand, and others on probability in Banach Spaces.

3.4 Recurrent Nets

One can ask similar approximation rate questions for recurrent nets. Recurrent nets provide universal identification models, in the sense that continuous- or discrete-time systems Σ of the type $\dot{x}$ [or x^+] $= f(x,u)\,, y = h(x)$ (under standard smoothness assumptions), can be approximately simulated by the behavior of a recurrent network. This was explained in [37]. By approximate simulation we mean as follows. In general, assume given two systems Σ and $\tilde{\Sigma}$ as above, where tildes denote data associated to the second system. and with same number of inputs and outputs (but possibly $\tilde{n} \neq n$). Suppose also given compact subsets $K_1 \subseteq \mathbb{R}^n$ and $K_2 \subseteq \mathbb{R}^m$, as well as an $\varepsilon > 0$ and a $T > 0$. Suppose further (this simplifies definitions, but can be relaxed) that for each initial state $x_0 \in K_1$ and each measurable control $u(\cdot) : [0,T] \to K_2$ the solution $\phi(t, x_0, u)$ is defined for all $t \in [0,T]$. The system $\tilde{\Sigma}$ *simulates* Σ *on the sets* K_1, K_2 *in time* T *and up to accuracy* ε if there exist two continuous mappings $\alpha : \mathbb{R}^{\tilde{n}} \to \mathbb{R}^n$ and $\beta : \mathbb{R}^n \to \mathbb{R}^{\tilde{n}}$ so that the following property holds: For each $x_0 \in K_1$ and each $u(\cdot) : [0,T] \to K_2$, denote $x(t) := \phi(t, x_0, u)$ and $\tilde{x}(t) := \tilde{\phi}(t, \beta(x_0), u)$; then this second function is defined for all $t \in [0,T]$, $\|x(t) - \alpha(\tilde{x}(t))\| < \varepsilon$, and $\left\|h(x(t)) - \tilde{h}(\tilde{x}(t))\right\| < \varepsilon$ for all such t. Assume that σ is a universal activation (dilates and translates of σ are dense on continuous functions with the compact-open topology). Then, *for each system* Σ *and for each* K_1, K_2, ε, T *as above, there is a* σ*-system* $\tilde{\Sigma}$ *that simulates* Σ *on the sets* K_1, K_2 *in time* T *and up to accuracy* ε. Thus, recurrent nets approximate a wide class of nonlinear plants. Note, however, that approximations are only valid on compact subsets of the state space and for finite time, so that many interesting dynamical characteristics are not reflected. This is analogous to the role of bilinear systems, shown to be universal in analogous fashion in Sussmann's well-known 1975 paper. As with bilinear systems, it is obvious that if one imposes extra stability assumptions ("fading memory" type) it will be possible to obtain global approximations, but this is probably not very useful, as stability is often a goal of control rather than an assumption.

3.5 Computing Risk Minimizer

Here we deal with question **Q3**: "Is it computationally feasible to find $\underline{\mathrm{Emp}}\,(\nu)$?", for hypotheses classes of the form $\mathcal{F} = \mathcal{F}_{\mathcal{A}}$. That is to say, given a cost function involving fitting an architecture to training data, what are the theoretical limitations of finding the value of the least error, and/or actually finding the minimizing parameters?

The approach we took in [20] originated with the work of Judd, Blum and Rivest, Lin and Vitter, and others. Judd observed that gradient descent techniques used in practice for the 1HL problem seem to be subject to a "curse of dimensionality". For the simpler case of linearly separable data, the perceptron algorithm and linear programming techniques help find a network – with no "hidden units" – relatively fast. Thus one may ask if there exists a *fundamental barrier* to training by general feedforward networks, a barrier that is insurmountable no matter which particular algorithm one uses. The simplest version of this is to ask about the tractability of the training problem, that is, of the question: "Can we determine if $\underline{\mathrm{Emp}}\,(\nu) = 0$? or in equivalent terms: "Given a network architecture (interconnection graph as well as choice of activation function) and a set of training examples, does there exist a set of weights so that the network produces the correct output for all examples?"

The simplest neural network, i.e., the perceptron, is a network that consists of one threshold neuron only. It is easily verified that the computational time of the loading problem in this case is polynomial in the size of the training set irrespective of whether the input is analog or binary. This can be achieved via a linear programming technique. We showed that, for 1HL networks employing a simple, piecewise linear activation function, and just two hidden units, the training problem is NP-complete. Recall that if one establishes that a problem is NP-complete then one has shown, in the standard way done in computer science, that the problem is at least as hard as most problems widely believed to be hard (the "traveling salesman" problem, Boolean satisfiability, and so forth). This shows that, indeed, *any possible* neural net learning algorithm based on fixed architectures faces severe computational barriers. Our results generalized those of Blum and Rivest, which only proved a similar NP-completeness conclusion for networks having the same architecture but differing from ours in that the activation functions are all of a hard threshold type.

4 References

1. Sussmann, H.J., "Real analytic desingularization and subanalytic sets: an elementary approach," *Trans. Amer. Math. Soc.* **317** (1990): 417-461.
2. Sussmann, H.J., "Uniqueness of the weights for minimal feedforward nets with a given input-output map," *Neural Networks*, **5** (1992): 589-593.
3. Sussmann, H.J., "On the use of neural networks in the analysis of nonlinear systems: realization, approximation, and feedback control," in *Proc. Congrès Satellite du Congrès Européen de Mathématiques on "Aspects Théoriques des Réseaux de Neurones, Paris, July 2-3 1992.*
4. Sussmann, H.J., and Y. Yang, "On the stabilizability of multiple integrators by means of bounded feedback controls," *Proc. 30th I.E.E.E. Conf. Decision and Control,* Brighton, UK (1991), pp. 70-72.

The following references all have Sontag as (co)author:

5. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, Springer, New York, 1990.
6. (With R. Alur and T.A. Henzinger) *Hybrid Systems III. Verification and Control*, Springer Verlag, Berlin, 1996.
7. (with H.J. Sussmann) "Backpropagation can give rise to spurious local minima even for networks without hidden layers," *Complex Systems* **3**(1989): 91-106.
8. "Sigmoids distinguish better than Heavisides," *Neural Computation* **1**(1989): 470-472.
9. (With H. Sussmann) "Backpropagation separates where perceptrons do," *Neural Networks*, **4(1991)**: 243-249.
10. "Feedback stabilization using two-hidden-layer nets," *IEEE Trans. Neural Networks* **3**(1992): 981-990.
11. "Feedforward nets for interpolation and classification," *J. Comp. Syst. Sci.* **45**(1992): 20-48.
12. (with H. Siegelmann) "Turing computability with neural nets," *Appl. Math. Lett.* **4(6)**(1991): 77-80.
13. (with R. Koplon) "Linear systems with sign-observations," *SIAM J. Control and Optimization* **31**(1993): 1245-1266.
14. (with F. Albertini) "For neural networks, function determines form," *Neural Networks* **6**(1993): 975-990.
15. (with F. Albertini) "State observability in recurrent neural networks," *Systems & Control Letters* **22**(1994): 235-244.
16. (with H. Siegelmann) "On the computational power of neural nets," *J. Comp. Syst. Sci.* **50**(1995): 132-150.
17. (with H. Siegelmann) "Analog computation, neural networks, and circuits," *Theor. Comp. Sci.* **131**(1994): 331-360.
18. (with H.J. Sussmann and Y. Yang) "A general result on the stabilization of linear systems using bounded controls," *IEEE Trans. Autom. Control* **39**(1994): 2411-2425.
19. (with R. Koplon and M.L.J. Hautus), "Observability of linear systems with saturated outputs," *Linear Algebra and Applics.* **205-206**(1994): 909-936.
20. (with B. DasGupta and H.T. Siegelmann) "On the complexity of training neural networks with continuous activation functions," *IEEE Trans. Neural Networks* **6**(1995): 1490-1504.
21. (with C. Darken, C., M. Donahue, and L. Gurvits) "Rates of convex approximation in non-Hilbert spaces," *Constructive Approximation* **13**(1997): 187-220

22. "Critical points for least-squares problems involving certain analytic functions, with applications to sigmoidal nets," *Advances in Computational Mathematics* (Special Issue on Neural Networks) 5(1996): 245-268.

23. (with R. Koplon) "Using Fourier-neural recurrent networks to fit sequential input/output data," *Neurocomputing* **15**(1997): 225-248.

24. (with B. Dasgupta) "Sample complexity for learning recurrent perceptron mappings," *IEEE Trans. Inform. Theory* **42** (1996): 1479-1487.

25. (with P. Koiran) "Neural networks with quadratic VC dimension," *J. Comp. Syst. Sci.* **54**(1997): 190-198.

26. "Shattering all sets of k points in 'general position' requires $(k-1)/2$ parameters," *Neural Computation* **9**(1997): 337-348.

27. (with P. Koiran) "Vapnik-Chervonenkis dimension of recurrent neural networks," *Discrete Applied Math.*, to appear.

28. (with P. Koiran) "Vapnik-Chervonenkis dimension of recurrent neural networks," in *Proc. Third European Conf. Computational Learning Theory*, Jerusalem, March 1997, to appear.

29. (with H.J. Sussmann and Y. Yang) "Global stabilization of linear discrete-time systems with bounded feedback," *Systems and Control Letters*, **30** (1997): 273-281.

30. (with H. Siegelmann and L. Giles) "The complexity of language recognition by neural networks," in *Algorithms, Software, Architecture* (J. van Leeuwen, ed), North Holland, Amsterdam, 1992, pp. 329-335.

31. (With F. Albertini and V. Maillot) "Uniqueness of weights for neural networks," in *Artificial Neural Networks for Speech and Vision* (R. Mammone, ed.), Chapman and Hall, London, 1993, pp. 115-125.

32. "Neural networks for control," in *Essays on Control: Perspectives in the Theory and its Applications* (H.L. Trentelman and J.C. Willems, eds.), Birkhauser, Boston, 1993, pp. 339-380.

33. (with B. DasGupta and H.T. Siegelmann) "On the Intractability of Loading Neural Networks," in *Theoretical Advances in Neural Computation and Learning* (Roychowdhury, V. P., Siu K. Y., and Orlitsky A., eds.), Kluwer Academic Publishers, 1994, pp. 357-389.

34. "Automata and neural networks," in *The Handbook of Brain Theory and Neural Networks*, M.A. Arbib, ed., M.I.T. Press, 1995.

35. "Spaces of observables in nonlinear control," in *Proc. Intern. Congress of Mathematicians 1994*, Volume 2, Birkhäuser Verlag, Basel, 1995, pp. 1532-1545.

36. "Interconnected automata and linear systems: A theoretical framework in discrete-time," in *Hybrid Systems III: Verification and Control* (R. Alur, T. Henzinger, and E.D. Sontag, eds.), Springer, NY, 1996, pp. 436-448.

37. "Neural nets as systems models and controllers," in *Proc. Seventh Yale Workshop on Adaptive and Learning Systems*, pp. 73-79, Yale University, 1992.

38. (With J.L. Balcázar, R. Gavaldà, and H. Siegelmann) "Some structural complexity aspects of neural computation," in *Proc. 8th Annual IEEE Conf. Structure in Complexity Theory*, San Diego, May 1993, pp. 253-265.

39. (With A. Macintyre) "Finiteness results for sigmoidal 'neural' networks," in *Proc. 25th Annual Symp. Theory Computing*, San Diego, May 1993, pp. 325-334.

40. (With F. Albertini) "Identifiability of discrete-time neural networks," *Proc. European Control Conference*, Groningen, June 1993, pp. 460-465.

41. (With F. Albertini) "Uniqueness of weights for recurrent nets," *Systems and Networks: Mathematical Theory and Applications*, Proc. MTNS '93, Vol. 2, Akademie Verlag, Regensburg, pp. 599-602.

42. (With H. Siegelmann) "Analog computation via neural networks," in *Proc. 2nd Israel Symposium on Theory of Computing and Systems (ISTCS93)*, IEEE Computer Society Press, 1993.

43. (with Y. Yang) "Stabilization with saturated actuators. a worked example: F-8 longitudinal flight control." *Proc. 1993 IEEE Conf. on Aerospace Control Systems*, Thousand Oaks, CA, May 1993, pp. 289-293.

44. (with C. Darken, C., M. Donahue, and L. Gurvits) "Rate of approximation results motivated by robust neural network learning," in *Proc. Sixth ACM Workshop on Computational Learning Theory*, Santa Cruz, July 1993.

45. (with F. Albertini) "State observability in recurrent neural networks," *Proc. IEEE Conf. Decision and Control, San Antonio, Dec. 1993*, IEEE Publications, 1993, pp. 3706-3707.

46. (with H.J. Sussmann and Y. Yang) "A general result on the stabilization of linear systems using bounded controls," *Proc. IEEE Conf. Decision and Control, San Antonio, Dec. 1993*, IEEE Publications, 1993, pp. 1802-1807.

47. (with R. Koplon) "Sign-linear systems as cascades of automata and continuous variable systems," *Proc. IEEE Conf. Decision and Control, San Antonio, Dec. 1993*, IEEE Publications, 1993, pp. 2290-2291.

48. (with B. DasGupta and H.T. Siegelmann) "On a learnability question associated to neural networks with continuous activations," *Proc. 7th ACM Conference on Learning Theory*, 1994, pp. 47-56.

49. (with R. Koplon) "Techniques for parameter reconstruction in Fourier-Neural recurrent networks," in *Proc. IEEE Conf. Decision and Control, Orlando, Dec. 1994*, IEEE Publications, 1994, pp. 213-218.

50. "Critical points for neural net least-squares problems," in *Proc. 1995 IEEE Internat. Conf. Neural Networks*, IEEE Publications, 1995, pp. 2949-2954.

51. "From linear to nonlinear: some complexity comparisons," *Proc. IEEE Conf. Decision and Control, New Orleans, Dec. 1995*, IEEE Publications, 1995, pp. 2916-2920.

52. (with B. Dasgupta) "Sample complexity for learning recurrent perceptron mappings," *Advances in Neural Information Processing Systems 8* (NIPS95) (D.S. Touretzky, M.C. Moser, and M.E. Hasselmo, eds.), MIT Press, Cambridge, MA, 1996, pp. 204-210.

53. (with P. Koiran) "Neural networks with quadratic VC dimension," *Advances in Neural Information Processing Systems 8* (NIPS95) (D.S. Touretzky, M.C. Moser, and M.E. Hasselmo, eds.), MIT Press, Cambridge, MA, 1996, pp. 197-203.

54. (with H.J. Sussmann) "Complete controllability of continuous-time recurrent neural networks," *Systems and Control Letters* **30**(1997): 177-183.